

中文用户手册

（未修订）

目录

第一部分 入门 1
摘要 1
1.1 简介 2
1.2 如何使用该手册 3
1.3 安装 Quest3D 4
1.4 实例场景 6
1.5 Quest3D 的用户界面 7
实例 11
1.6 Channel 16
实例 23
1.7 程序流程 26
实例 29
1.8 模版 30
1.9 三维场景 31
实例 31
1.10 发布 35
实例 36
1.11 小结 39
第二部分 虚拟场景 40
摘要 40
2.1 三维物体 41
实例 45
2.2 动画 47
实例 49
2.3 导入物体 53
实例 53
2.4 表面属性 56
实例 62
2.5 光照和阴影 65
实例 67
2.6 像机 70
实例 71
2.7 图形用户接口 74
实例 75
2.8 声音 77
实例 78
2.9 地形和环境 80
实例 83
2.10 粒子系统 86
实例 87
2.11 角色动画 89
实例 92
第三部分：编程 96
摘要 96
3.1 逻辑 97
实例 100
3.2 数学 104
实例 108
3.3 For Loop 111
实例 114
3.4 数组 117
实例 121
3.5 Multiple channel groups 124
实例 127
3.6 Mathematical operators(数学操作) 129
实例 131
3.7 Pathfinding（寻径） 137

实例 138
3.8 有限状态机 142
实例 144
第四部分 高级 147
摘要 147
4.1 项目管理 148
实例 149
4.2 物理仿真 151
实例 154
4.3 数据库连接 158
实例 160
4.4 网络 164
实例 166
4.5 Lua 脚本 170
实例 172
附录 176
A1 从 Max 和 Maya 中导入 176
A2 用户接口（略） 179
A3 快捷键 179

第一部分 入门

摘要

1.1：简介

该部分介绍了 Quest3D 的基本情况和相关信息。

1.2：如何使用该手册

介绍了该手册的组织方式和结构。

1.3：安装

介绍了 Quest3D 的安装流程。

1.4：实例

使用安装包中的实例来演示 Quest3D 的一些特性。

1.5：用户界面

提供使用该程序的指南。同时介绍了一个新的概念 “channel”，channel 是 Quest3D 的基本模块。

1.6：通道

channel 是 Quest3D 工程的核心，这里介绍了它的基本属性。

1.7：程序流程

介绍了 Quest3D 工程的执行方式和实时更新的特性。

1.8：模版

模版是预先定义好的 channel 或 channel 组。这里给出了 Quest3D 中能够使用的模版列表。

1.9：三维场景

虚拟场景由多种元素组成，例如三维模型，摄像机和灯光。

1.10：发布

介绍如何将 Quest3D 的项目保存为一个独立运行的程序。

1.11：小结

回顾该部分的内容

1.1 简介

目前，三维图像正在迅速影响整个世界。在好莱坞的电影中可以完全展示活的怪兽和整个行星。电脑游戏可以提供奇幻的角色在广袤的世界中进行交互式的探险。商业和科学的一些展示可以在三维场景中实现。甚至整个训练课程都可以在虚拟现实中完成。

尽管三维图像已经成为主流，但是制作它们还需要花费很大的精力。只有使用一些适当的专用工具才可能取得令人折服的效果。

Quest3D 是一个完美的交互式软件开发包。使用它你可以完成产品展示、建筑可视化、虚拟训练和计算机游戏。

Quest3D 使用了一种独特编程方式，在这种方式下你不需要编写大量的复杂代码，开发人员可以使用大量具有强大功能的模块来实现其特定的功能。这些模块不仅易于使用而且是可扩展的。

使用 Quest3D 意味着开发的实时性：你可以直接修改最后的结果，而不需要编译或渲染图像。

Queste3D 拥有一些非常好的特性的集合。大量生动的人物，漂亮的植物，阴影，火和烟的效果并且可以非常容易的在场景中添加水的效果。高级特性包括物理模拟，路径发现，数据库连接和网络支持。

该手册是 Quest3D 的指南，该手册将提供理论和实际的例子。阅读完该手册你应该能够创建你可以想象的任何交互式场景。

欢迎进入 Quest3D 的奇妙世界！

1.2 如何使用该手册

下面描述了该手册的组织方式和结构。

部分

本手册共分为四个部分

- ③ 第一部分："入门"介绍 Quest3D。
- ③ 第二部分："虚拟场景"详细讨论三维模型和动画。
- ③ 第三部分："编程"介绍 Quest3D 工程的核心——"代码"。
- ③ 第四部分："高级"介绍一些高级特性，其中的某些特性只在特定的版本中可用。

章节

每个章节都开始于一个新的主题的介绍，然后将讨论与之相关的 Queset3D 的通道和功能。最后，给出一个详细实例使用户能够亲手试验本章介绍的主题。

实例

安装完 Quest3D 之后，你可以在你的硬盘上找到本手册中所介绍的每一个实例的完整场景，缺省情况下这些实例位于目录：

C:\Program Files\Act-3D\Quest3D 3.0\Tutorials\

格式

本手册使用如下的通用格式：

- ③ 新出现的概念将使用黑体表示。例如：Quest3D 的组建模块被称为 '**Channel**'
- ③ 已经在 Quest3D 程序中使用过的通道将使用单引号标示，并且首字母大写。例如：将'Logic'channel 连接到'Start Scene'channel
- ③ 来自于模版列表中的 channel 或模版将使用斜体表示，并且首字母将大写。例如：添加一个 *Value*channel 到你的程序中；
- ③ 模版列表中 channel 和模版的位置将作为文件夹使用反斜线分割。例如：Logic\Channel Caller
- ③ 菜单选项将使用如下方式表示。例如：'File>Save Group As'
- ③ 文件和目录使用如下方式。例如 C:\Program Files\Act-3D\Quest3D 3.0\
- ③ Quest3D 目录的子文件夹使用如下方式表示。例如：..\Resources\Textures\

参考手册

作为该书的一个补充，参考手册中包含的 Quest3D 中每一个 channel 的描述。你可以在程序运行时按 F1 键来查看该参考手册。

1.3 安装 Quest3D

下面给出 Quest3D 的安装步骤：

③ 启动计算机。
③ 将 Quest3D 的安装光盘放入光驱。
③ Quest3D 将自动启动并询问是否要安装 Quest3D。



如果 Windows 没有自动开始：

③ 在 Windows 桌面上双击我的电脑图标。



③ 双击 CD-ROM 盘符。
③ 双击” Autorun.exe”。
③ 稍等片刻后出现一个窗口询问是否要安装 Quest3D。

Quest3D 安装（继续）

③ 选择”Install Quest3D”。
③ 单击”Next”按钮继续安装。
③ 默认的安装目录为 C:\Program Files\\Act-3D\Quest3D 3.0\
③ 建议使用默认的安装目录。使用’Next’按钮确认安装目录。
③ 建议在桌面上创建一个快捷方式。按’Next’继续安装。
③ 单击’Install’按钮。Quest3D 将拷贝文件到你的计算机上，这将需要一段时间。
③ 单击’finish’完成安装。

Quest3D 依赖于微软公司的 DirectX 扩展库。该库中包含一些高级的图形和声音功能。因此你必须安装 DirextX9.0c 或更高版本才能使用 Quest3D。

安装 DirectX

- ③ 从 Quest3D 的安装菜单中选择’Install DirectX9.0’选项。
- ③ 选择你使用的操作系统和语言类型。
- ③ 单击’Yes’以确认安装。
- ③ 如果你接受协议单击’Yes’。
- ③ DirectX 将开始安装。

启动 Quest3D

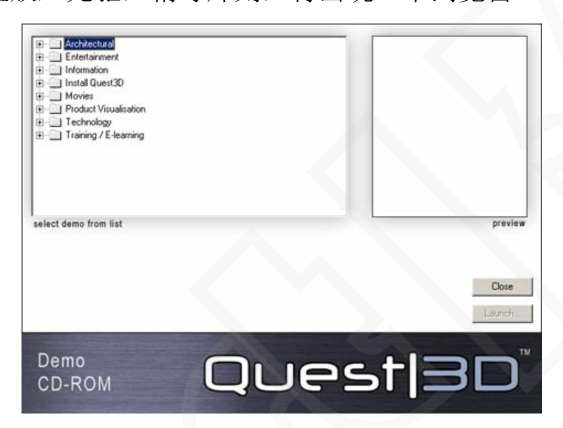
- ③ 单击 Windows 开始菜单。
- ③ 从开始菜单中选择程序>Act-3D>Quest3D 3.0>Quest3D 3.0
- ③ 输入你的 Quest3D 序列号。
- ③ 单击出现的 Quest3D 的闪屏以继续。

1.4 实例场景

在开始使用 Quest3D 之前请先浏览一下安装包中附带的已完成的场景。它们展示了 Quest3D 的一些特性。

查看实例场景

将 Quest3D 的光盘放入光驱，稍等片刻，将出现一个浏览窗口。



如果浏览窗口没有自动出现：

③ 双击“我的电脑”

③ 双击 CD-ROM.

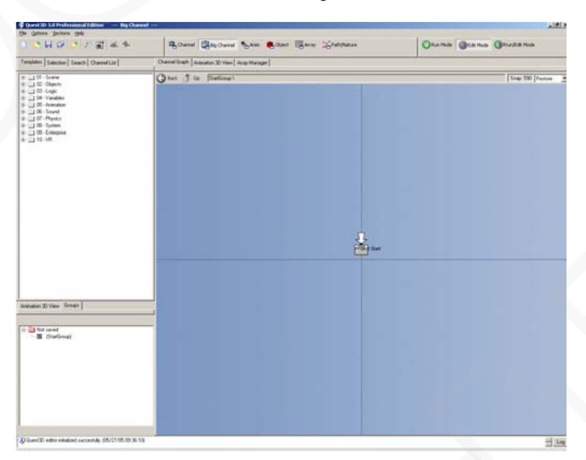
③ 双击’Autorun’或’Autorun.exe 文件。片刻后浏览窗口将出现。

> 在查看这些实例的时候，确定它们都使用了哪些特性，在这些特性中哪些可能会用在你的工程中。

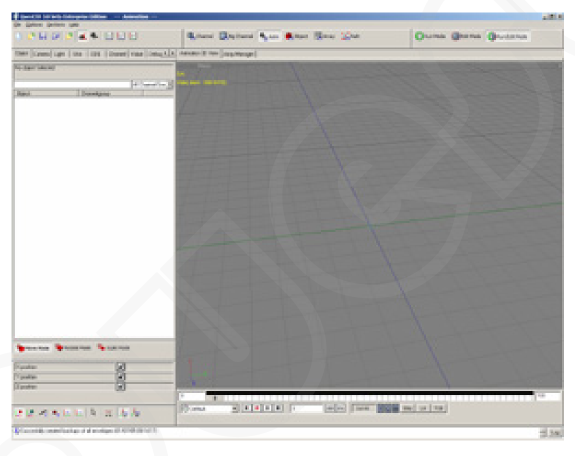
本手册将帮助你创建一个类似的场景，或者更加完善。

1.5 Quest3D 的用户界面

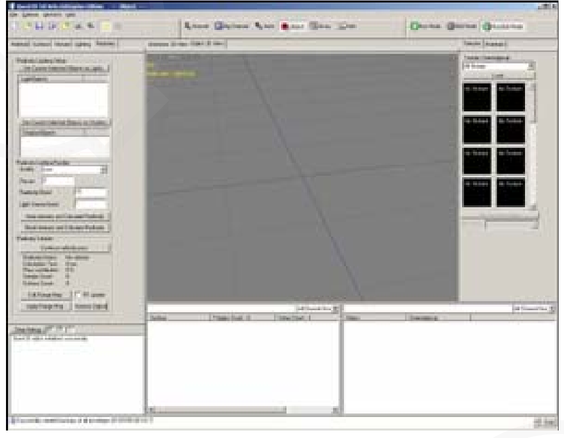
Quest3D 提供了功能强大的特性的集合。为了组织他们将这些特性划分成了不同的 ”**Sections**”。下面将介绍 Quest3D 中三个最重要的部分。



Channels Section 是 Quest3D 的核心部分。Quest3D 在启动之后即显示 Channels Section，在 Channel Section 是创建一个工程最基础部分。



Animation Section 是可使三维模型、相机和灯光进行定位和运动的部分。该部分中包含大量可用于测试工程的预览窗口。



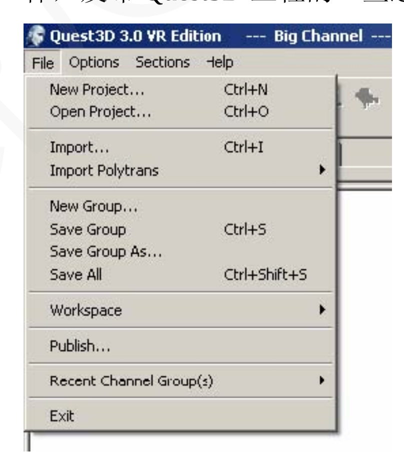
Object Section 是一些备用和候选的对象。Quest3D 提供大量的选项来处理物体表面属性例如颜色和纹理。

注意由于 Quest3D 的基本界面可以改变，因此不同用户的 Quest3D 看起来可能会不太一样，在该手册中我们使用的是标准的界面。你可以参看参考手册中的“自定义”一章以获得更多信息。

本章仅仅讨论 Channels Section。Animation 和 Object Sections 将在本手册的第二部分：“虚拟场景”中介绍。

File 菜单

File 菜单提供了用于打开和保存工程的选项。此外，还提供了从其他程序中导入三维物体，发布 Quest3D 工程的一些选项。发布工程的过程将在下一章中讨论。



“**New Project**”选项将清除所有先前加载的 channel 和 channel 组并打开一个新的工作空间。在接下来的一些指导中你可能会想使用该选项来清空工程。使用该选项的效果与退出并重新启动 Quest3D 相同。

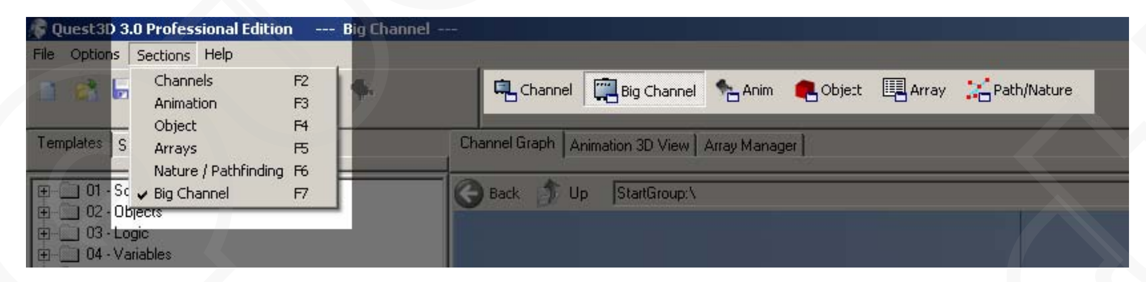
“**Open Project**”允许你加载一个 Channel 组或者一个工程文件

使用“**Save Group As**”可以将一个 Channel 组保存为一个新的文件。如果你想覆盖现有的文件可以使用”**Save Group**”选项。使用”**Save All**”选项可以保存整个工程。

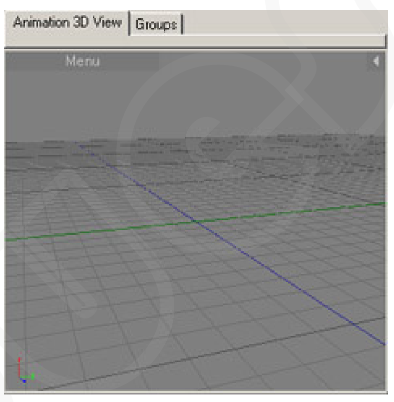
注意对于新手来说使用”Save All”选项是保存文件最安全的方式

Section 导航

可以使用各种方法在 Quese3D 的 Section 中进行切换。最常用的方法是使用**菜单**或是**工具栏**上的 **Section 选择按钮**进行切换。



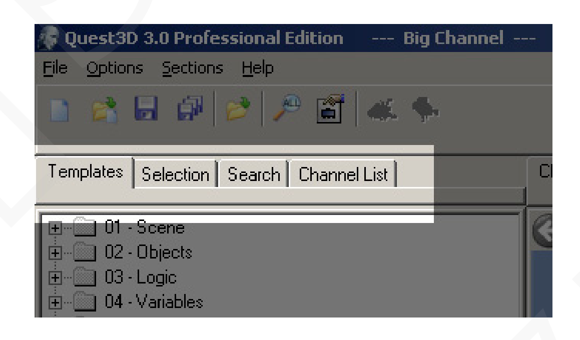
Section 中的界面元素



在 Quest3D 中某个窗口或按钮可能会出现在不同的 Section 中。例如，Channel Section 中包含一个 **Animation 3D 视口**。这是一个与 Animation Section 相同类型的窗口。这个 Animation 3D 视口显示了该工程的实时预览效果。修改 Channel 图后工程的最终效果将立即显示在 Animation 3D 视口中。

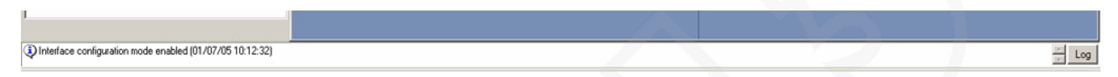
标签

Quest3D 的许多窗口都使用了标签控件来切换。在 Channel Section 中最重要的标签是”**Template list**”, “**Channel list**”和”**Search window**”.可以通过单击不同的标签来访问这些窗口。



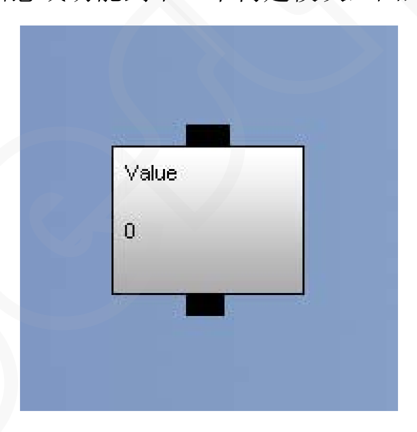
日志栏

日志栏位于 Quest3D 的最下端。日志栏显示一些常见信息和特定问题。可以使用”Log”按钮来查看从程序启动到现在的所有历史记录。



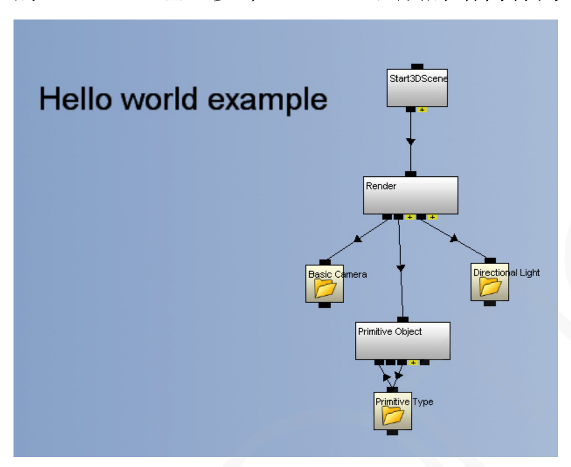
Channel 视图

Channel 视图中将显示的当前正在构建的 Channel 的结构。Quest3D 程序都是使用构建模块来完成的。这些构建模块被称为”**channel**”。下图显示了一个 Quest3D 中的 channel。Quest3D 的构建模块”引导”信息或功能到下一个构建模块，因此被称为”channel”



Channel 组

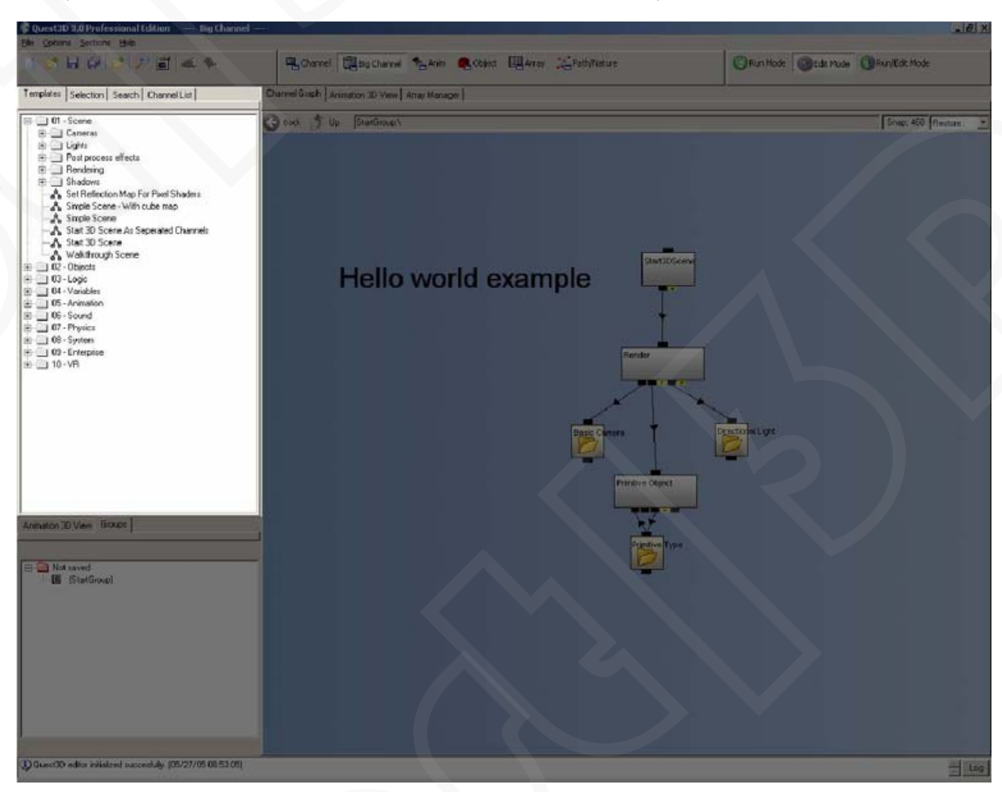
下图显示了一个小的 **Channel 组**。多个 channel 组成的结构称为 Channel 组。



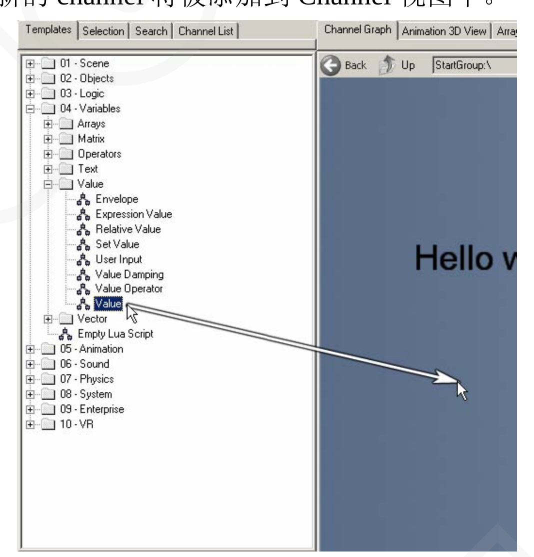
上图同时显示了用户的注释（文本"Hello world example"）。此外，某些构建模块被放置在文件夹中。

添加 Channel

在 Quest3D 的左边的**模版列表**中包含了所有可用的 Quest3D 构建模块。



可以通过从模版列表中拖动一个 channel 的到 Channel 视图中的方式来添加一个新的 channel。"**拖动**"表示按住鼠标左键并移动鼠标到一个新的位置，并释放鼠标左键。
释放鼠标左键后，一个新的 channel 将被添加到 Channel 视图中。

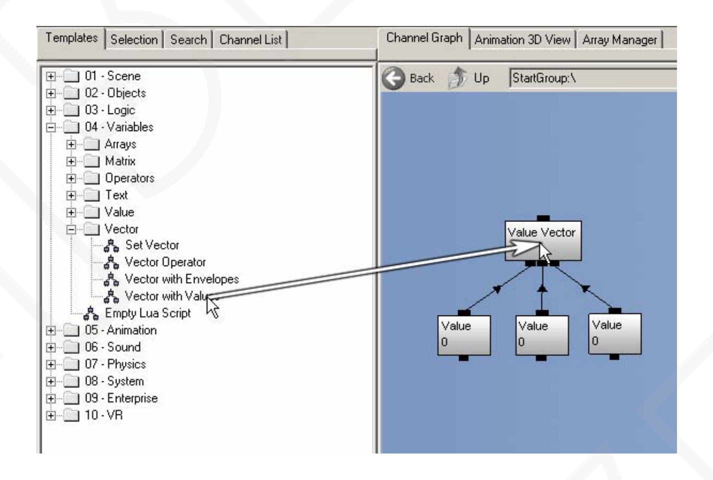


Template（模版）

Template 是一个预先定义好的 channel 或 channel 组。使用模版能够极大地提高你的工作效率。添加模版就像添加一个 channel 一样简单。.

在下面的例子中，一个 *Vector* 模版被添加到 Channel 视图中。一个 *Vector* 模版包含四个

channel。



标准用户行为

大多是 Quest3D 的用户界面的工作方式与其他 Windows 软件的工作方式相同。下表描述了大量重要的 Quest3D 的行为。下表中的术语将被用在整个手册中

移动鼠标到特定位置	移动屏幕上的指针到屏幕上的特定位置。
单击	按下鼠标左键然后松开
双击	按下鼠标左键快速松开然后在快速按下并释放
拖动鼠标	按住鼠标左键，然后移动鼠标
从A拖动到B	按住鼠标左键，并移动鼠标到特定位置
选择一项	当鼠标经过一个项目时按下鼠标左键并释放

实例

下面的实例将介绍 Quest3D 的用户界面。同时还会涉及到 channel（构建模块）。该实例的目的是熟悉 Quest3D 的用户界面。

启动 Quest3D:

③ 单击 Windows 开始菜单。

③ 从开始菜单中依次选择程序>Act-3D>Quest3D 3.0>Quest3D 3.0

③ 在出现的 Quest3D 闪屏上单击以进入程序。

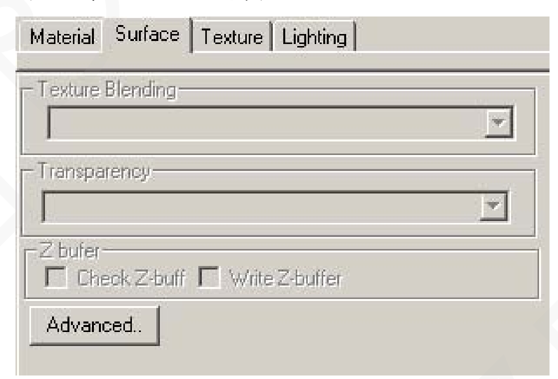
Step by step:

③ 默认情况下，Quest3D 将打开一个 Channel Section。Channel Section 是使用”channel”来创建程序的地方。

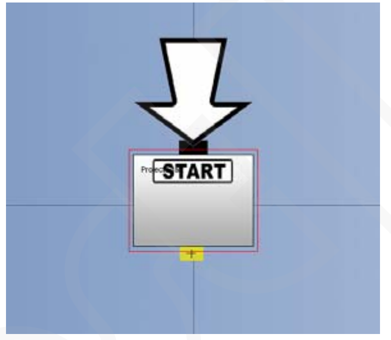
③ 从菜单中选择 Section>Animation 切换到 Animation Section。

③ 使用菜单切换到其他 Section。

③ 按下工具栏上适当的按钮以切换到其他 Section。
③ 进入 Object Section。
③ 单击 Surface 标签以显示 Surface 选项。



③ 返回 Material 标签。
③ 进入 **Big Channel Section**。
③ 屏幕中最大的蓝色背景的区域被称为 Channel 视图。该区域中包含了组成你程序的所有 channel。目前，该区域总仅仅包含一个名为”Project Start”的 channel。



③ 当你的鼠标在 Channel 视图上时，你可以向前或向后滚动鼠标滚轮来放大或缩小 Channel 视图。

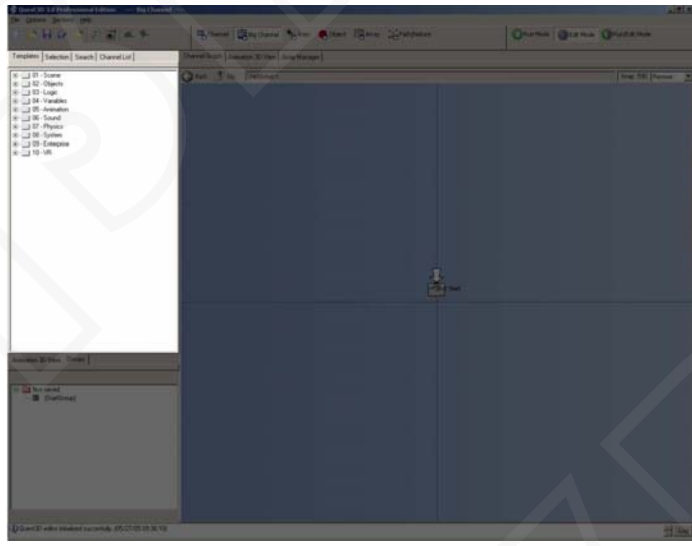
如果你的鼠标没有滚轮，你可以按住键盘上的”Alt”键并按住鼠标右键然后前后移动来发放大或缩小。

放大 Channel 视图可以更加清楚地查看一个 channel。而缩小 Channel 视图可以得到一个工程的概况。

③ 将鼠标放置在 Channel 视图中。按住鼠标中键移动鼠标可以移动 Channel 视图。

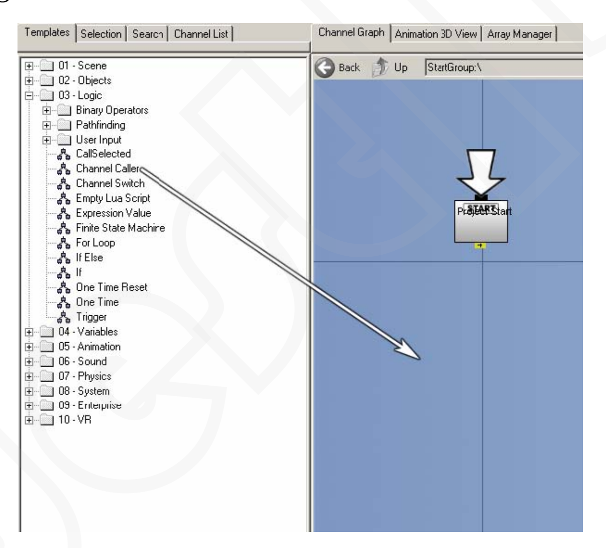
如果没有鼠标中键：使用'Alt + 鼠标左键'来移动 Channel 视图。

③ 查看屏幕左侧的 Template 列表。该列表包含了大量 channel 和具有各种功能的 channel 的组合

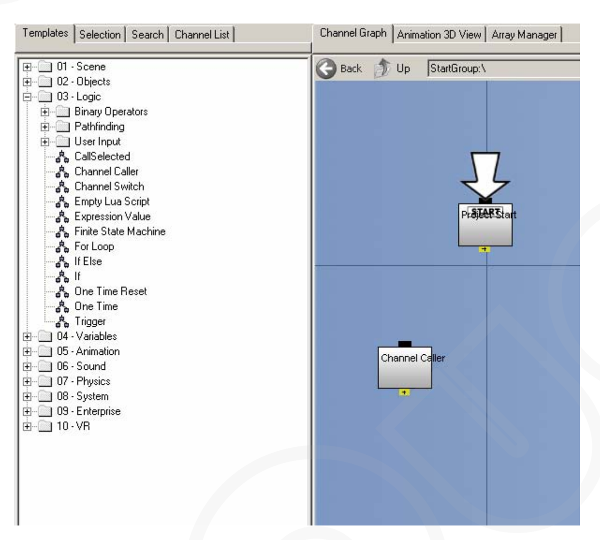


③ 单击'Logic'左侧的'+'号。注意该项目展开显示一系列项目。

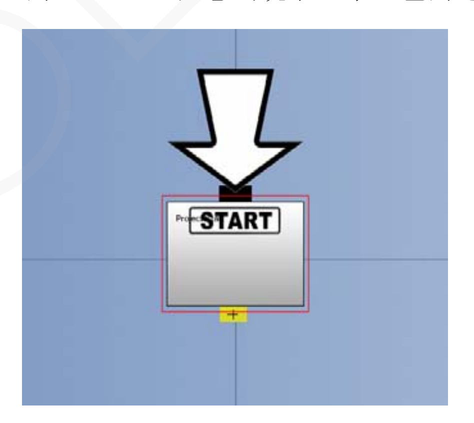
③ 移动鼠标到'Channel Caller'（一个 *Channel Caller* 可以在你的工程中创建一个逻辑路径。例如一个名为 Project Start 的 *Channel Caller* 可调用其它名为'Render Scene'和'Do User Logic'的 channel）。



③ 单击并按住鼠标左键，移动鼠标到屏幕中间名为'Project Start'的 channel 下方，释放鼠标左键。这样就将一个 channel 拖动到 Channel 视图中。

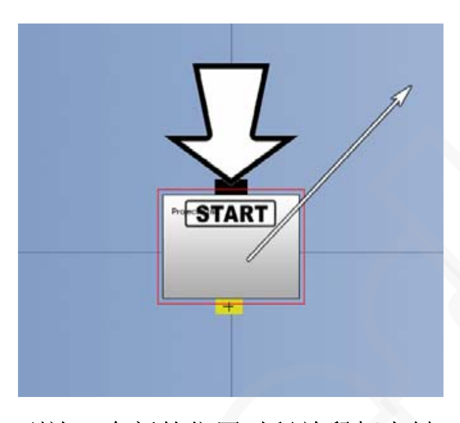


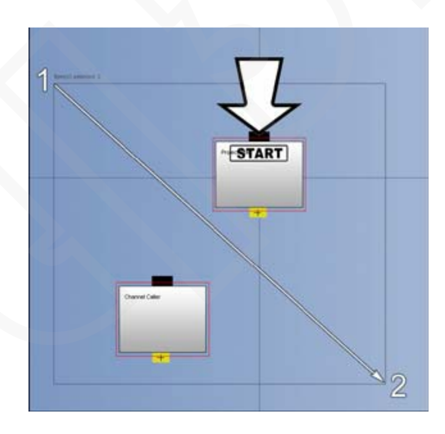
③ 单击名为'Project Start'的 channel。注意出现了一个红色的选择轮廓线。



③ 在蓝色的背景上单击以取消选择名为'Project Start'的 channel。注意红色的选择轮廓线消失了。

③ 将鼠标指向名为'Project Start'的 channel。按住鼠标左键并移动鼠标可以移动该 channel。

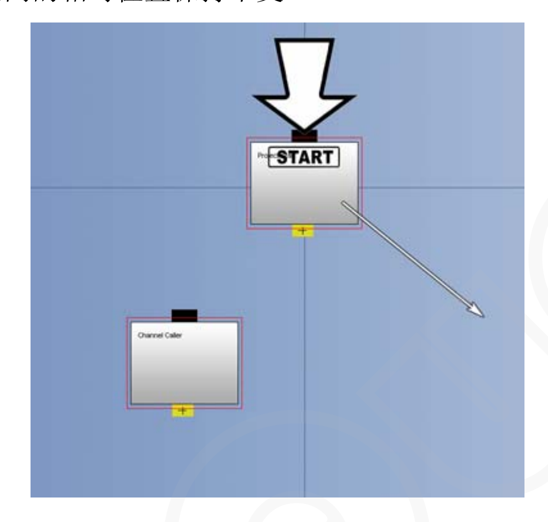




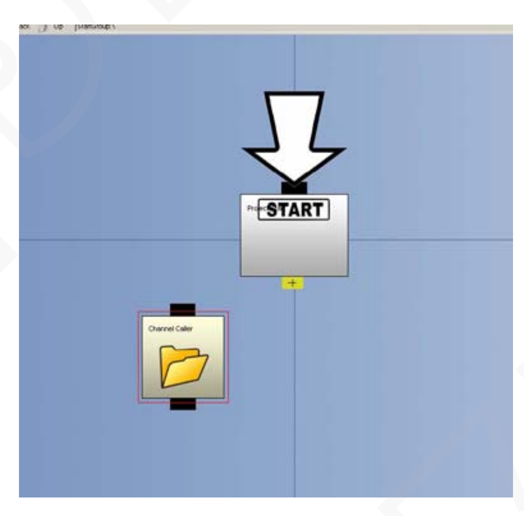
③ 到达一个新的位置时释放鼠标左键。注意该 channel 被移动了。

③ 可以从 Channel 视图的左上角开始拖动鼠标到你想选择的所有 channel 的右下角，并释放鼠标左键来框选多个 channel。图中鼠标左键在位置'1'处被按下并在位置'2'处被释放，该过程产生了一个选择框，在该框中的所有 channel 都将被选中。选中的 channel 将出现红色的轮廓线。

③ 如果要同时移动多个 channel，首先选中你想移动的 channel。然后，移动鼠标到任何选中的 channel 上。按住鼠标左键并移动鼠标，这时所有选中的 channel 都将被移动，而它们之间的相对位置保持不变。



③ 选择名为'Channel Caller'的 channel 并按空格键。该 channel 将被存放在一个文件夹中。



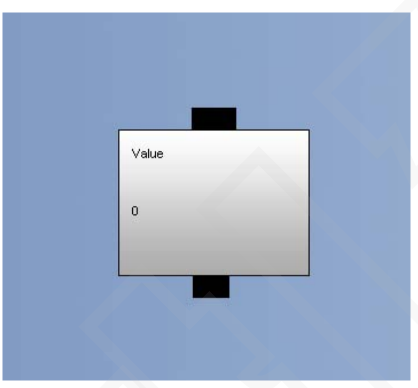
文件夹将被命名为创建文件夹时鼠标所指向的 channel 的名称。

③ 选择文件夹并按下空格可以解开该文件夹。
③ 选择名为'Channel Caller'的 channel 并使用键盘上的'Delete'删除它。

1.6 Channel

Quest3D 程序由若干构建模块组成。这些构建模块被称为 **channel**，每个 channel 都有一个特定的功能。

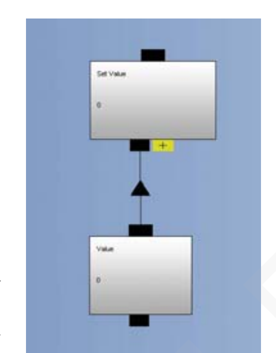
在 Quest3D 中，channel 是一个矩形模块。下图中的 channel 被称为一个 *Value* channel。它可以存储一个数值。



Channel 的上方和下方的黑色矩形块被称为**连接块**。可以通过 channel 上的连接块来连接多个 Channel。

右图中，上方的 channel 被称为**父 channel**。下方的 channel 被称为**子 channel**。子 channel 通常被用作父 channel 的输入或输出数据。

没有连接子 channel 的输入连接块不会接受输入数据。通常情况下，空的输入连接块被设置为 0。



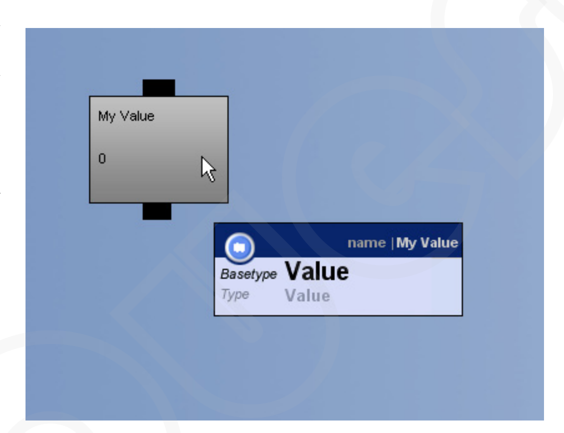
Channel 组

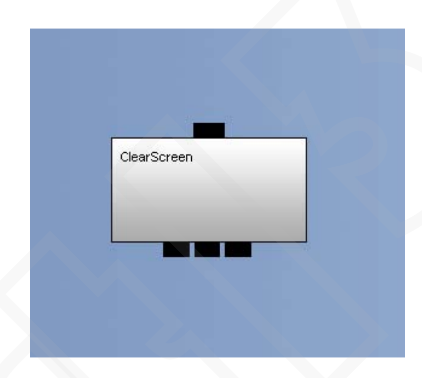
一个通过连接多个 channel 构成的结构被称为 **Channel 组**。一个 Quest3D 工程由一个或多个 Channel 组组成。例如，一个组在场景中显示一个房间而另一个组显示一个角色。这两个 Channel 组组合起来显示一个在房间中的角色。

Channel 信息

可以将鼠标移动到一个 channel 上来查看该 channel 的信息。稍等片刻后将出现一个弹出式窗口。

右图中显示了一个名为'My Value'的 channel。除名称以外，每一个 channel 都有一个 **Basetype（基类型）**和一个 **Type（类型）**属性。一个 *Value* channel 的 Basetype 为 *Value*，Type 也为 *Value*。





左图中的 channel 名为'Clear Screen'。

这个 channel 用来改变该工程的背景色。计算机程序中的颜色通常由红，绿，蓝三个部分组成。因此，颜色通常被存储在一个 **RGB** 值的集合中。

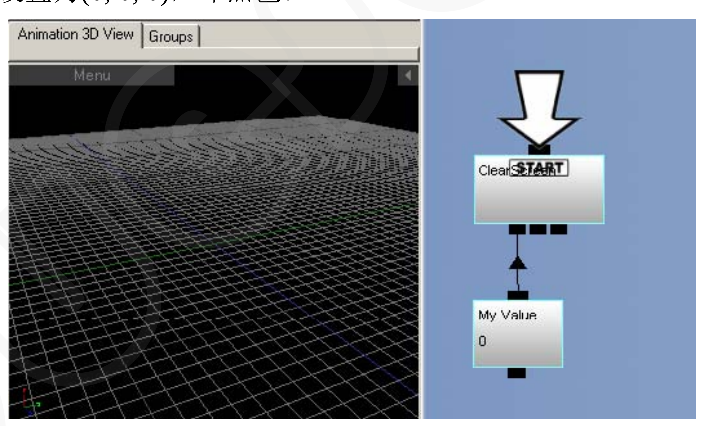
该 channel 接受三个类型为 *Value* 的子 channel:分别代表 RGB 三个颜色。

查看连接块相关信息的方法与查看 channel 信息的方法相同。弹出窗口将显示连接块的名称和 Basetype（基类型）。

下图中'Clear Screen' channel 的第一个连接块被称为'R'。它代表了该 channel 的颜色中红色部分。

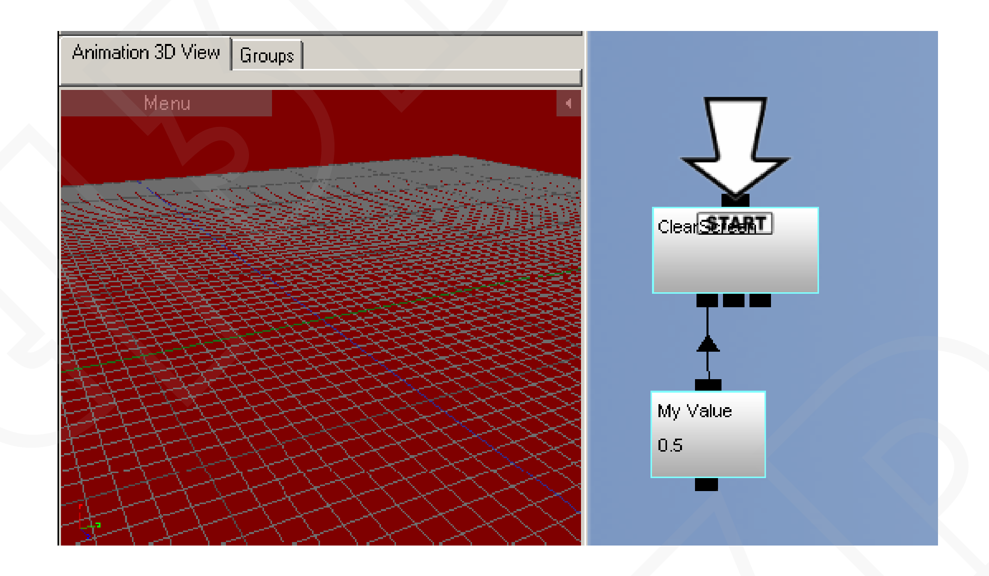


信息窗口也显示了第一个连接块只接受 Basetype 为 *Value* 的 channel。因此名为'My Value'的 channel 可以作为该 channel 的子 channel。在下图中名为'My Value'的 channel 被赋值为 0。'Clear Screen' channel 的第二个和第三个连接块为空，因此也被赋值为 0。那么该'Clear Screen' channe 的颜色被设置为(0, 0, 0)，即黑色。



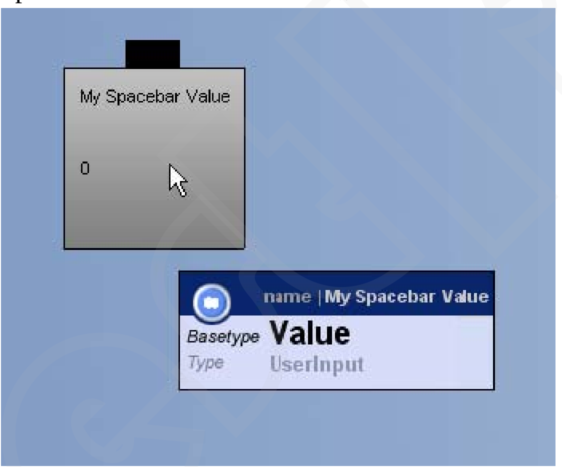
注意网格只会出现在 Quest3D 的编辑器中，而不会出现在能够独立运行的程序中。网格是 Quest3D 的助手项

下图显示了'My Value' channel 被设置为 0.5 后，背景颜色变成了红色。



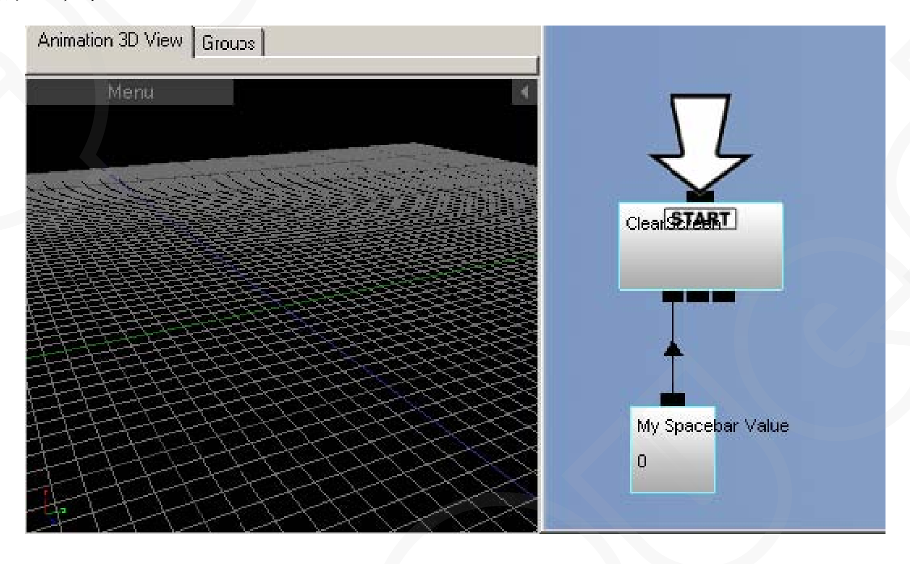
用户输入

下图显示了名为'My Spacebar Value'的 channel 的信息。

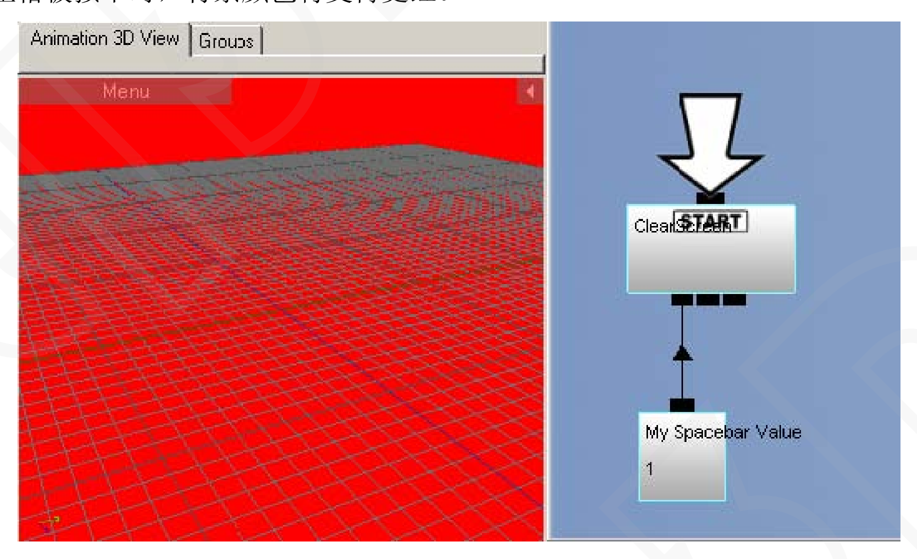


该 channel 的 Basetype 属性同样为 *Value*。然而，它的 Type 为 *User Input*。一个 *User Input* channel 可以将许多不同的用户活动转化为数值，例如，键盘，鼠标和手柄的输入。

由于'My Spacebar Value' channel 的 Basetype 为 *Value*，因此它可以作为 *Clear Screen* channel 的一个子 channel。

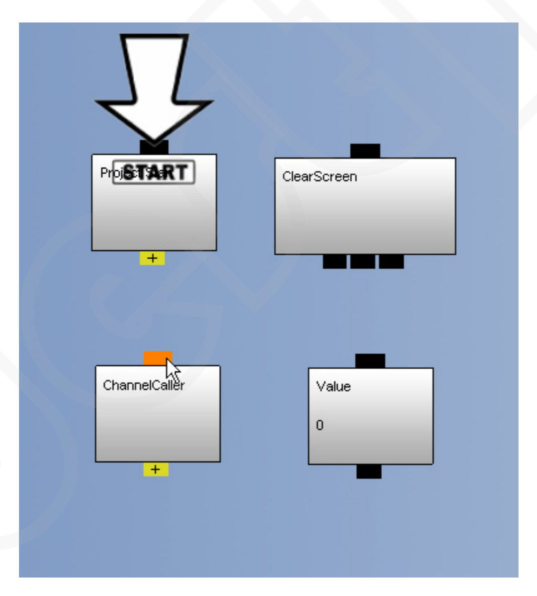


上图和下图中显示了'Clear Screen' channel 根据用户的输入将背景颜色从黑色改变为红色。当空格被按下时，背景颜色将变得更红。



多重连接块

下图中显示四个不同 channel：两个 *Channel Caller*，一个 *Value* 和一个 *Clear Screen* channel。



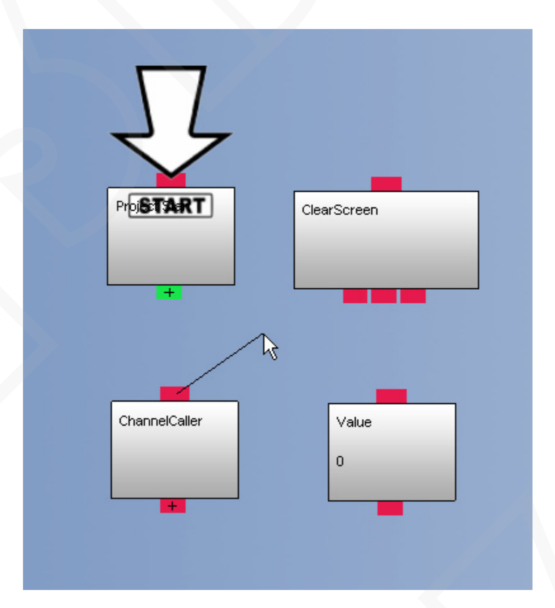
不同的 channel 可以有不同数量和类型的连接块。例如，*Clear Screen* channel 接受三个子 channel 而 *Value* channel 接受一个子 channel。

带有'+'号的黄色连接块表明该连接块可以连接多个子 channel。当一个子 channel 连接到该连接块上时，'+'号将向右移动，同时创建一个新的位置以接受另一个子 channel。

连接

移动鼠标到'Channnel caller' channel 顶部的连接块，这时该连接块变为橙色。

在下图中，通过按住鼠标左键，一个连接动作从'Channel Caller' channel 开始连接到另外的 channel

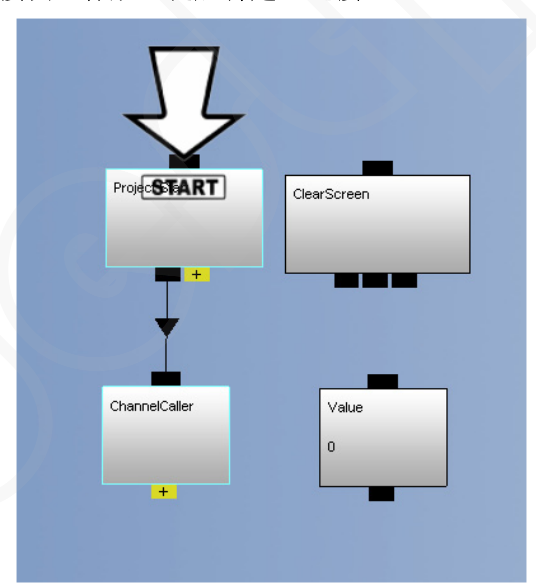


在连接过程中所有的连接块都将改变颜色为红色或者绿色。只有绿色的连接块可以被连接，而红色的不能。

上图中，一个连接从 *Channel Caller* channel 开始，'Project start' channel 的一个连接块为绿色表示该连接块可以接受该连接。

由于 *Clear Screen* channel 和 *Value* channel 只接受 *Values*。因此，它们的连接块是红色。

如果连接可用，在连接块上释放左键后将建立连接。



除了连接到连接块以外，你还可以在 channel 上释放连接，该 channel 将自动连接到目标 channel 的一个开启并且可用连接块上

快速连接：

③ 选择一个子 channel。

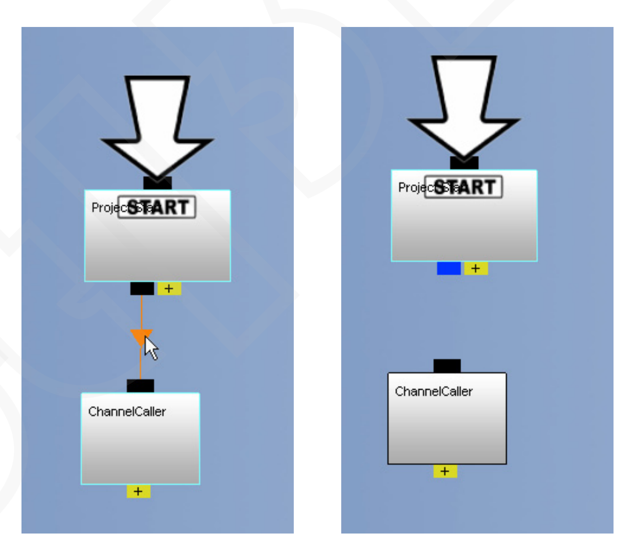
③ 按下键盘上的'L'键。

③ 单击父 channel。

删除连接

在连接上单击鼠标左键即可选中该连接。选中后可以使用键盘上的'Delete'键删除该连

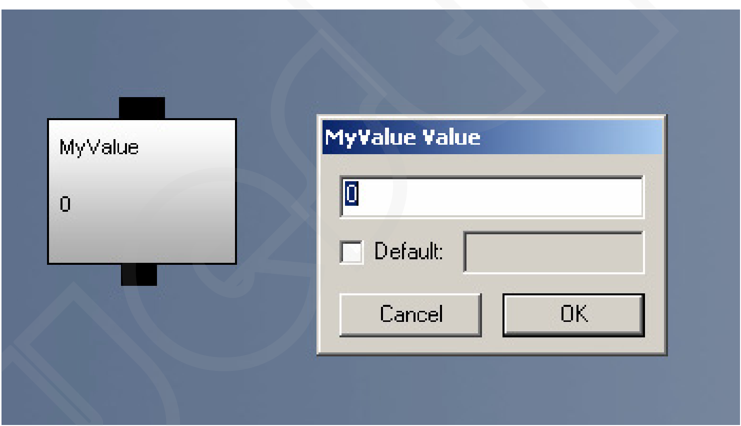
接。



删除一个连接后，将留下一个空的蓝色连接块。Channel 将忽略这些空的连接块。如果你连接一个新的子 channel 到蓝色的连接块上时，这个空的位置将被填充。

Channel 属性对话框

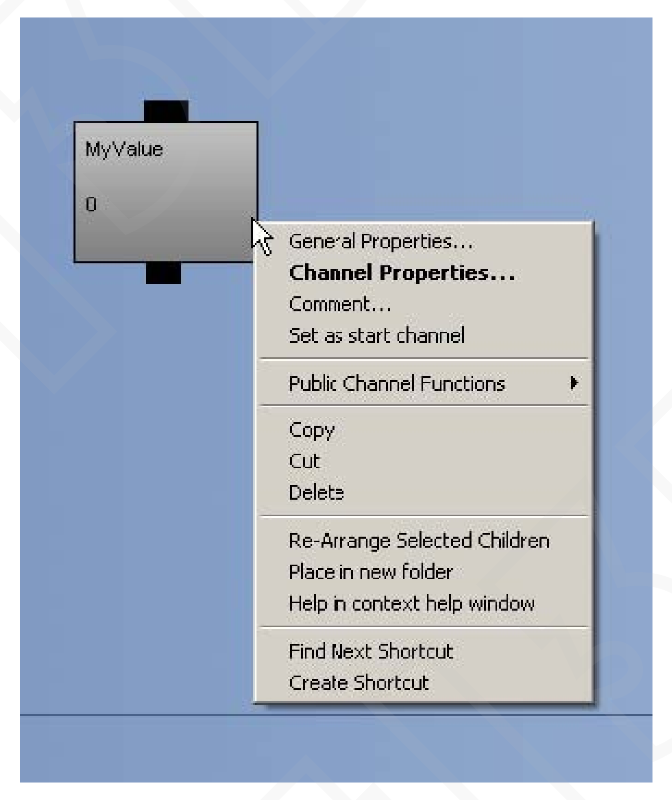
双击一个 channal 将打开这个 channel 特有的**属性对话框**下图显示的是属于 *Value* channel 的属性对话框。



在该属性窗口中你可以输入一个新的数值。注意打开一个对话框时 Quest3D 不能暂停，也就是说当你的程序正在运行的时候你可以改变该数值，改变后的结果将立即显示出来。

上下文菜单

右键单击 channel 将打开**上下文菜单**。



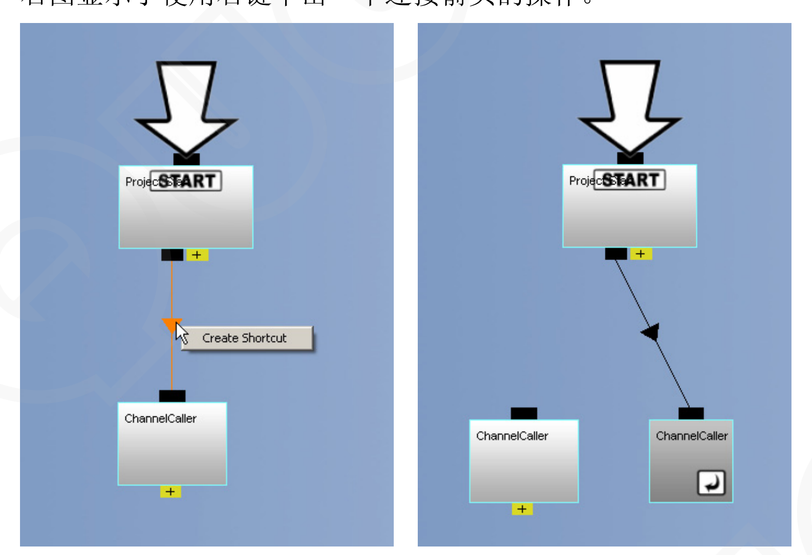
Channel 的名称可以在'General Properties'选项中修改。选择'Channel Properties'将打开 channel 属性对话框。

快捷键

Quest3D 只能指定 channel 的快捷键，而不能指定文件夹的快捷键。

Channel 的快捷键可以通过使用 channel 的上下文菜单来创建。同样还可以通过右键单击两个 channel 之间连接上的箭头来设置快捷键。

右图显示了使用右键单击一个连接箭头的操作。



使用快捷键可以直接引用到最初的 channel。使用快捷键可以使 Channel 组更易读。

使用文件夹和快捷键以确保你的工程清晰易读。

实例

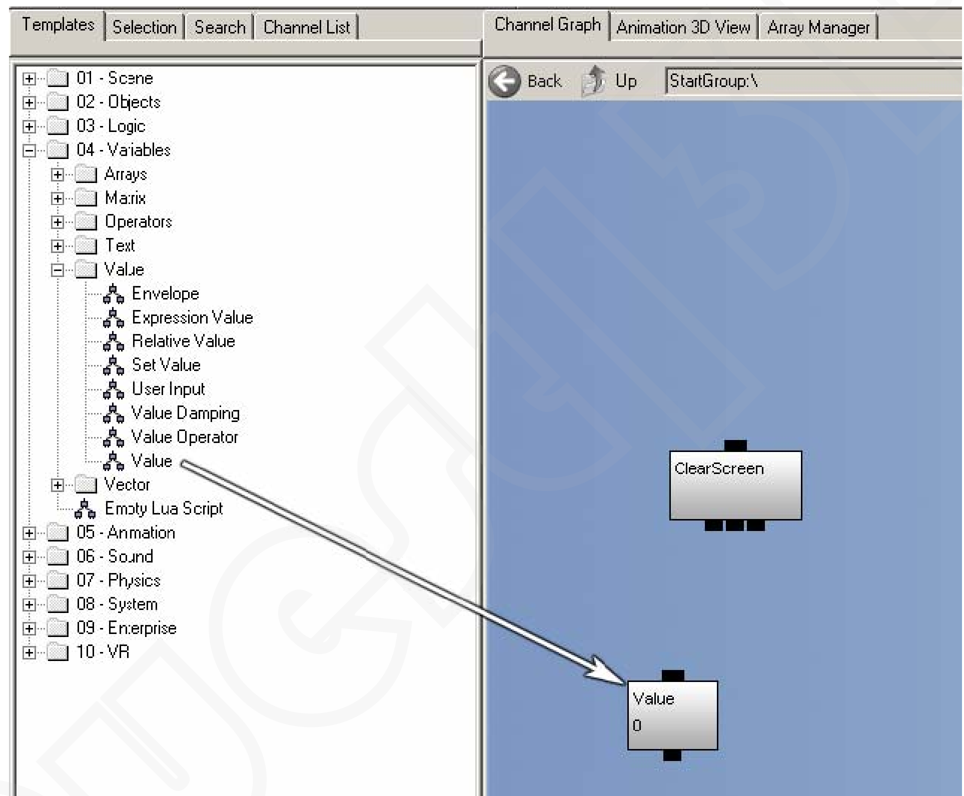
下面的实例将介绍 channel 属性对话框。阅读完该实例后，你能够找到和理解两个 channel 的连接块之间 Type 和 Basetype 的含义

Step by step:

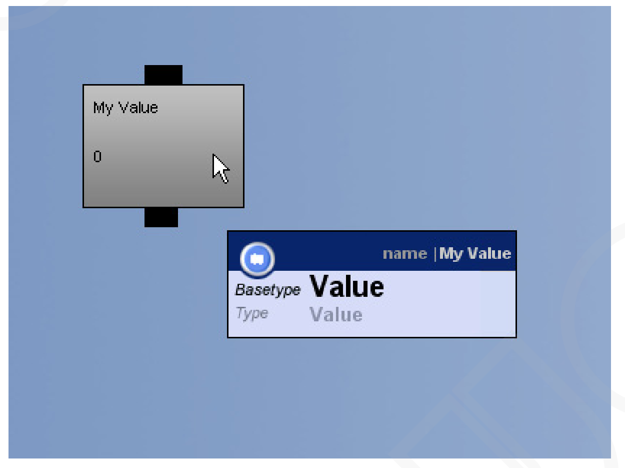
③ 启动 Quest3D。默认情况下 Quest3D 将打开 Channel Section。

③ 从 Template 列表中拖动如下 channel 到 Channel 视图中：System\Clear Screen。

③ 添加如下 channel：Variables\Value\Value。



③ 移动鼠标到'Value' channel 并等待信息弹出窗口的出现。

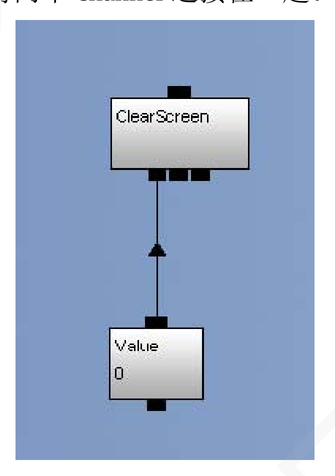


③ 移动鼠标到'Value' channel 上方的连接块上，注意黑色的连接块变为橙色。

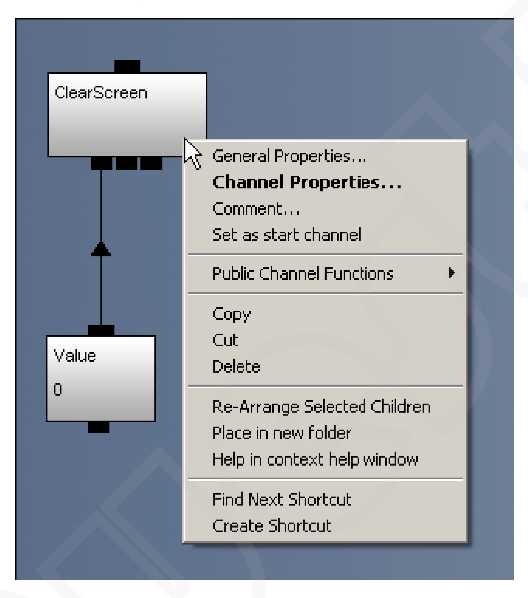
③ 按住鼠标左键，并移动到'Clear Screen' channel 上。**先不要释放鼠标**，注意：一条

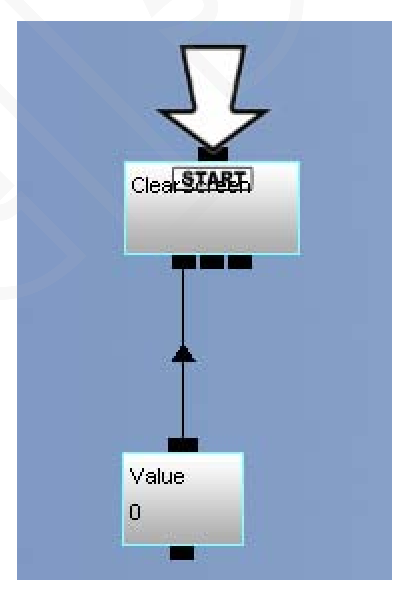
黑色的线从'Value' channel 顶端连到鼠标的当前位置。

③ 释放鼠标左键。注意：一条黑色的线从'Value' channel 的顶端连接到'Clear Screen' channel 的底部，这样就将两个 channel 连接在一起。



③ 移动鼠标到'Clear Screen' channel 上并单击右键以打开上下文菜单。选择'Set as start channel'选贤。

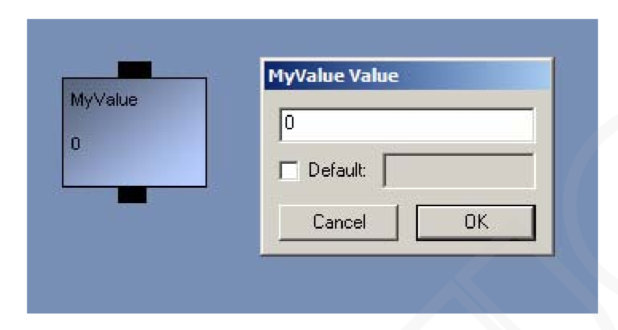




③ 右图显示了在'Clear Screen' channel 上出现了一个大的箭头符号，该符号表示该 channel 是一个开始 channel。

③ 单击屏幕左下角的'Animation 3D View'标签以确保 Animation 3D 视口可见。

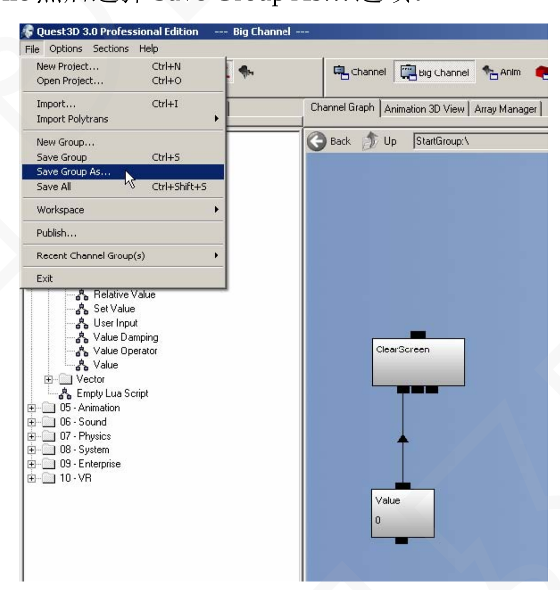
③ 双击'Value' channel 打开它的属性对话框。



③ 在文本框中输入一个 0 到 1 之间的值。注意：这个操作将影响 Animation 3D 视口中的背景颜色。

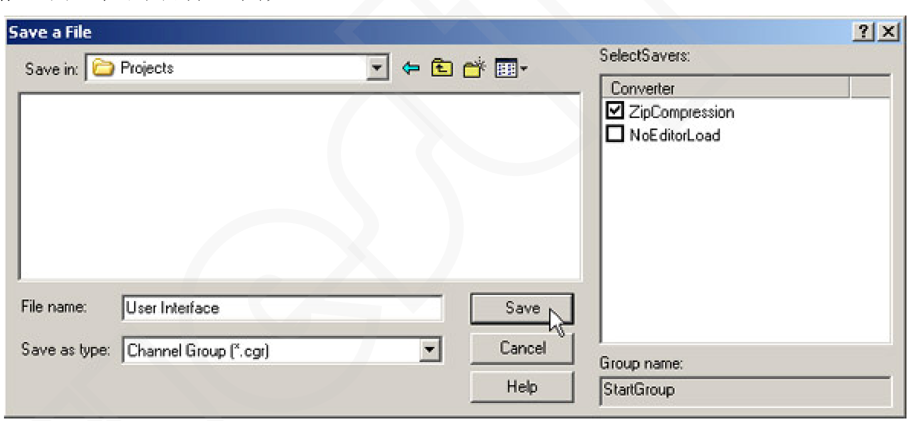
③ 单击'OK'按钮接受并关闭该对话框。

③ 在菜单中选择'File'然后选择'Save Group As…'选项。



③ 找到一个适当的位置存储该工程，例如：C:\Program Files\Act-3D\Quest3D 3.0\Projects\

③ 输入该工程的名称，例如'Introduction to channels'



③ 单击'Save'按钮保存。

Quest3D 程序的文件夹下有一个程序的例子，下面给出了它的目录和名称。正如我们在 1.2 章如何使用本手册中介绍的一样，'..\'代表如下的文件夹：C:\Program Files\Act-3D\Quest3D 3.0\

③ 从菜单中选择'File'并选择'Open Project...'选项。

③ 找到如下目录下的'Finished scene'。完整的路径为 C:\Program Files\Act-3D\Quest3D 3.0\Tutorials\1.6 -Channels\

③ 打开文件'Channels - Complete.cgr'

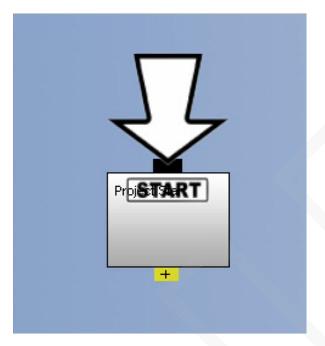
③ 分析这个完成的场景看看你是否能够理解该实例。

完成后的场景：

③ ..\Tutorials\1.6 -Channels\Channels - Complete.cgr

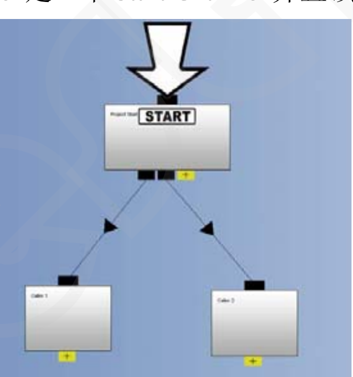
1.7 程序流程

一个 Quest3D 程序从一个 Start Channel 开始。一个 Start Channel 使用一个大的向下的箭头来标示。



在 channel 上下文菜单中的'Set as start channel'选项可以指定 Quest3D 程序从该 channel 开始运行。

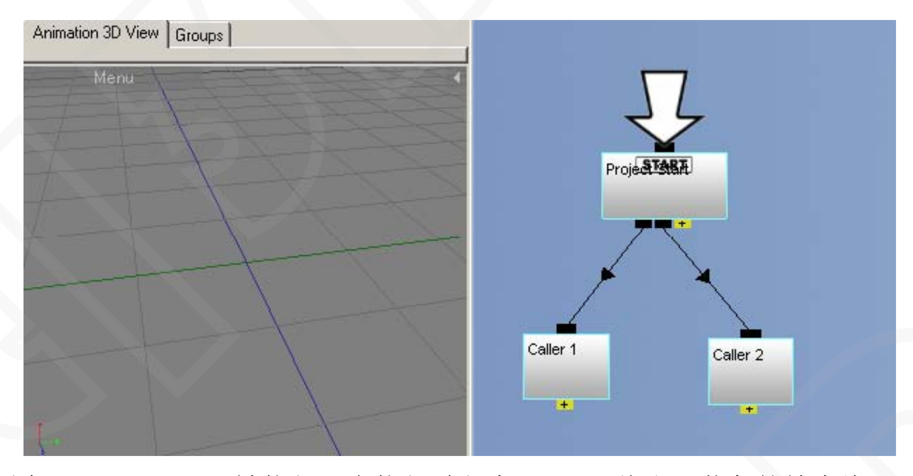
下图中，'Channel Caller' channel 是一个 Start Channel 并且该 channel 拥有两个子 channel。



Channel Caller 是工程流程的基础

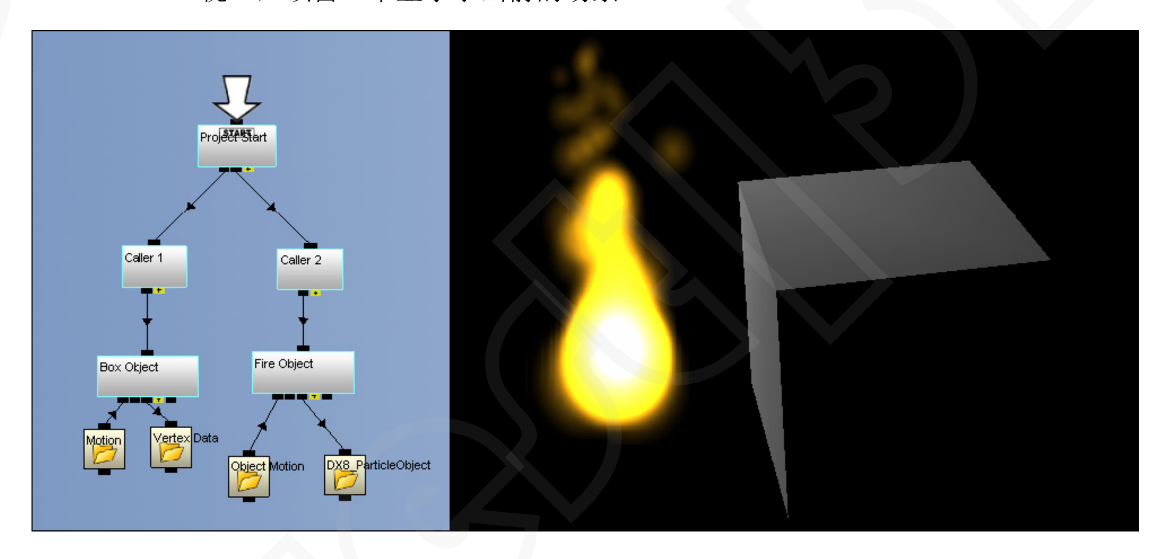
激活工程

只有 Animation 3D 视口打开的时候 Quest3D 的工程才能被激活执行。



上图中，Start Channel 被执行，在执行过程中 channel 将出现蓝色的轮廓线。

下图中显示了一个 Quest3D 场景。左侧显示的是 channel 的结构。右侧显示的是 Animation 3D 视口，该窗口中显示了当前的场景。



帧

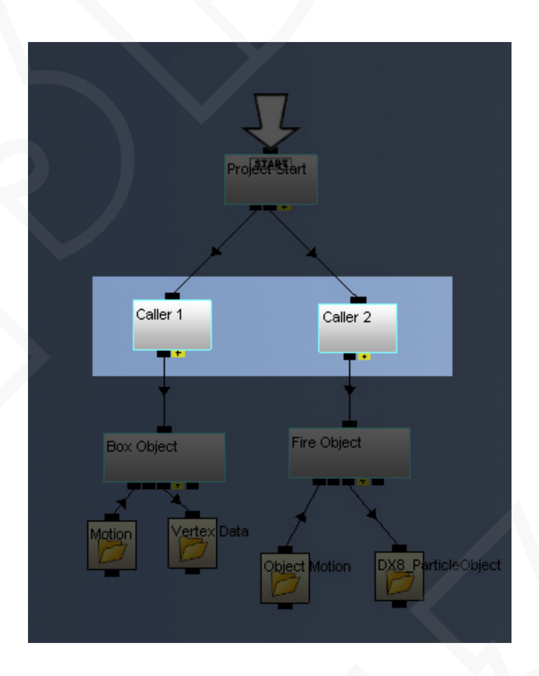
Quest3D 使用实时的工作方式。也就是说它不断的执行工程并更新预览显示。一个完整的 channel 结构的执行过程被称为**帧**。而计算所有结果的过程被称为**渲染**。

帧率表示每秒程序执行的次数。该值依赖于工程的复杂度和用于渲染场景的计算机硬件

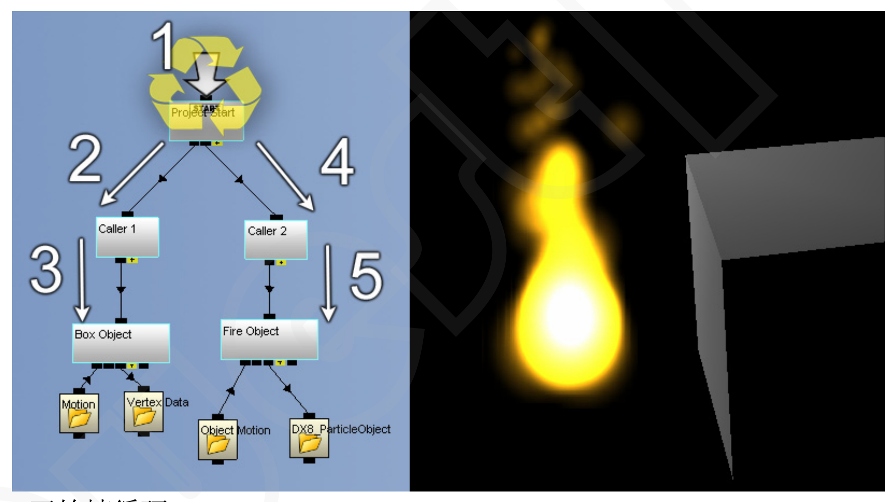
程序流

在 Quest3D 中**调用**一个 channel 意味着实现一次它的所有功能，而这些功能基于所有的从子 channel 接收的输入。调用一个 channel 的结果依赖于它的 Type 属性。

一个 Channel Caller 按照从左向右的顺序调用它的子 channel。在下图中，'Caller 1' channel 首先被调用，在屏幕上显示一个正方体。然后调用'Caller 2' channel，渲染一团火。



在下图中显示了 Channel 组的流程。



1. 开始帧循环
2. Channel Callers 从左向右执行，因此 Start Channel 首先调用左侧名为'Caller 1'的 channel。
3. 'Caller 1' channel 再次调用它下面的层次结构。该层次结构开始于一个名为'Box Object'的'3D Object'。这个 channel 将在屏幕上显示一个立方体，如上图中右侧部分。
4. 执行完层次结构中左侧的部分后，该 Start Channel 继续执行它的第二个子 channel 名为'Caller 2'。
5. 'Caller 2' channel 调用它的名为'Fire Object'的子 channel，该 channel 和它的子 channel 将创建一个火，如上图所示。

当该结构被完整地执行之后，程序将开始一轮新的执行。

实例

下面的实例将讨论 Quest3D 的程序执行流程。

Quest3D 场景：

- ..\Tutorials\1.7 - Program flow\Program flow.cgr

Step by step:

- 启动 Quest3D。
- 打开文件'Program flow.cgr'。该文件位于上述路径下。该文件包含一个简单的场景，其中有一个立方体和一团火。
- 单击屏幕左下角适当的标签以打开 Animation 3D 视口。
- 查看 Animation 3D 视口中的三维场景。注意：立方体和火都被渲染。
- 右键单击'Caller 1' channel 从上下文菜单中选择'Set as start channel'选项。
- 查看 Animation 3D 视口中的三维场景。注意：只有火可见。
- 右键单击'Caller 2' channel 从上下文菜单中选择'Set as start channel'选项。
- Examine the scene in the Animation 3D View window. Note only the box object is visible.
- 查看 Animation 3D 视口中的三维场景。注意：只有立方体可见。

完成后的场景:

- ..\Tutorials\1.7 - Program flow\Program flow - Complete.cgr

1.8 模版

（Template）模版是一个预定义的 channel 或 channel 组，使用模版可以提高工作效率。

模版可以被直接用于场景中，使用模版的好处是你不需要立即知道模版中各个部分的含义。稍后，当你对 Quest3D 比较熟悉的时候，你可以详细看看各种类型的模版。你甚至可以构建你自己的模版。

下面给出的是 Quest3D 中模版的列表。该表将使你对模版集合有一个大致的了解。本手册下面的部分将大量的讨论这些模版并详细的介绍它们的功能。

分类	描述	模版实例
Scene	三维场景组织	Render；Camera; Light
Objects	三维物体，表面和特效	Primitive Object; Texture; Particle System; Pixel Shader ; Vertex Shader
Logic	条件和触发	Channel Caller; If ; Trigger; Channel Switch User Input; Finite State Machine; Motionplanning
Variables	数值和操作符	Value ; Vector ; Matrix Text ; Expression Value
Animation	定时器，规则和角色动画	Timer Value ; Timer Command ; Motionset
Sound	声音	Wav File ; Mp3 File ; Sound Command
Physics	物理模拟 (基于开放的动态引擎)	ODE Body ; ODE Joint ; ODE Command
Network	连接多台计算机	Network Value ; Network Text Network Actions
Database	连接到数据库	DB Source ; DB Query ; DB Value
Dynamic Loading	程序的动态加载部分	Save Channel to Buffer ; Array Buffer Remove Group
Windows	弹出式窗口	Message Box ; File Dialog ; Open URL
Viewers	工程浏览	ActiveX Event ; WinAmp Waveform
System	系统信息和命令	System Info ; System Command
VR	虚拟现实支持	Glove

1.9 三维场景

所有的 Quest3D 程序都使用相同的规则。本章介绍如何使用 Quest3D 创建一个工程。

Quest3D 场景

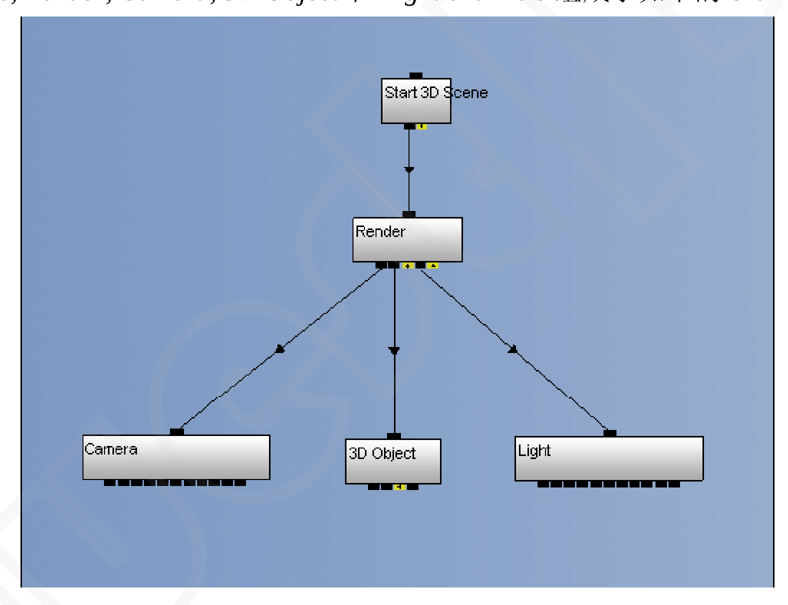
三维模型在大多数的 Quest3D 程序中起着非常重要的作用。三维物体可以从你所熟悉的三维建模软件中导入。而且还可以使用来自于 Quest3D 中的三维物体

如同电影一样，一个 Quest3D 场景需要一个相机和若干灯光。

Render channel 用于在场景中显示三维物体。

Start 3D Scene channel 用于初始化一个三维场景。同时处理分辨率和背景颜色。

Start 3D Scene, *Render*, *Camera*, *3D Object* 和 *Light channels* 组成了如下的 channel 结构。



实例

在本实例中你将创建一个简单的 Quest3D 程序。该程序的场景中有一个三维物体，一个相机和一个灯。

在本章中，将开始使用模版。在该实例中使用到的模版将在下面“需要的模版”中给出。

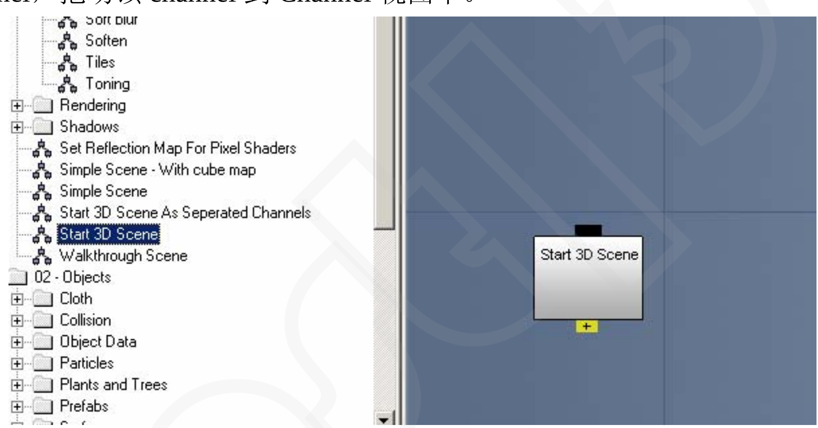
本实例的第一个目标是熟悉 Quest3D 程序的基本结构。第二个目标是练习向程序中添加一些 channel 并连接它们。

需要的模版：

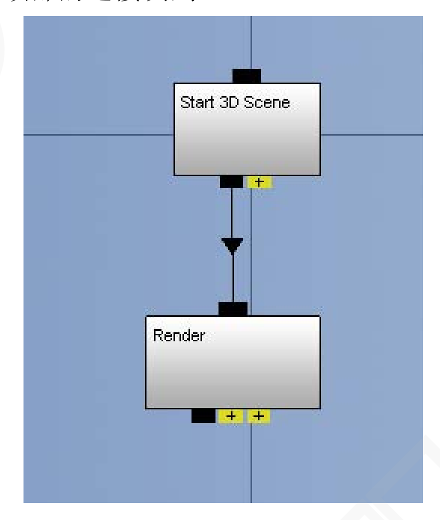
- ③ Scene \Start 3D Scene
- ③ Scene \ Rendering \ Render
- ③ Scene \ Cameras \ Object Inspection Camera
- ③ Objects \ Primitive Object
- ③ Scene \ Lights \ Point Light

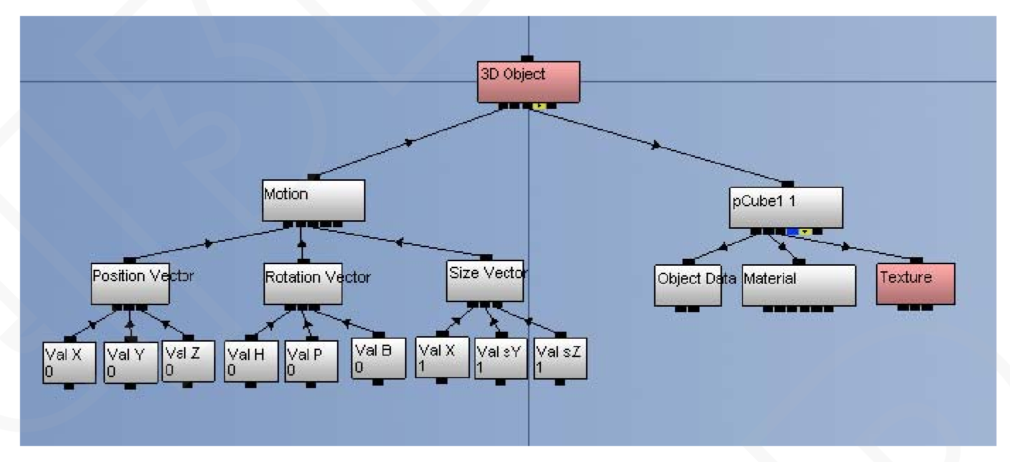
Step by step:

- ③ 启动 Quest3D。
- ③ 从菜单中选择'File > New Project...'以创建一个新的工程，并起名为'First project scene'。
- ③ 在 Template 列表中，找到 Start 3D Scene 模版。在上面“需要的模版”中可以找到它的位置: 3D Items \ Start 3D Scene。
- ③ 在模版列表中单击'3D Items'前方的'+'，展开该项。模版列表中有一个 Start 3D Scene channel，拖动该 channel 到 Channel 视图中。



- ③ 右键单击'Start 3D Scene' channel 从菜单中选择'Set as start channel'。
- ③ 在模版列表中找到 *Render* channel。
- ③ 拖动 Render channel 到 Channel 视图中，将它放置在'Start 3D Scene' channel 的下方。
- ③ 连接'Render' channel 顶部的连接块到'Start 3D Scene' channel 下方的子连接块。





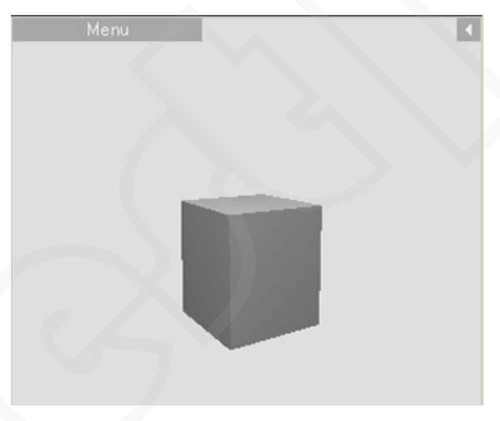
③ 拖动一个 Object Inspection Camera 道 Channel 视图中。

③ 连接'Object Inspection Camera' channel 到'Render' channel 的第一个子连接快上。

③ 去顶左下角的 Animation 3D 视口显示了使用投影相机（Project Camera）后的场景。

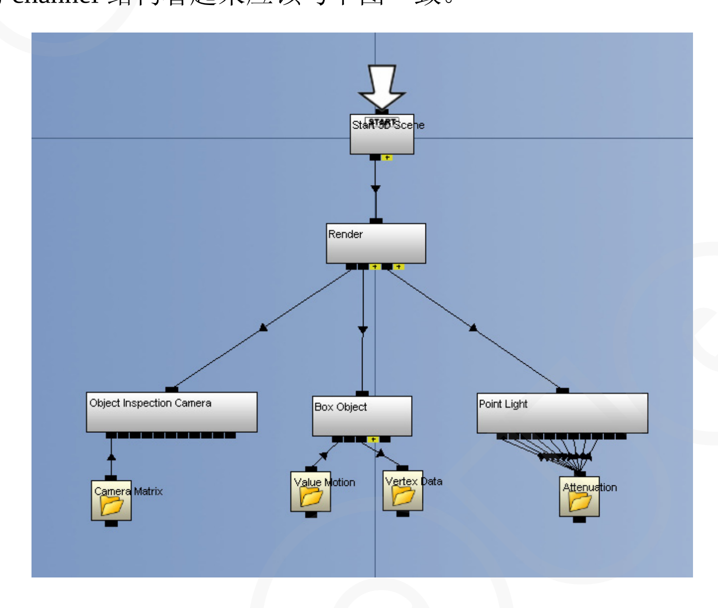
投影相机的图标位于工具栏上。

③ 添加一个 Primitive Object 模版并将'3D Object' channel 连接到'Render' channel 的第二个子连接块上。注意：此时在 Animation 3D 视口中三维物体将显示出来。



③ 拖动一个 *Point Light* 模版到 Channel 视图中。将'Scene Light' channel 连接到'Render' channel 的最后一个子连接块上。注意：汽车受到了光源的影响。

③ 现在你的 channel 结构看起来应该与下图一致。



- ③ 从菜单中选择'File > Save Group As...'。
- ③ 选择 C:\Program Files\Act-3D\Quest3D 3.0\Projects\文件夹。
- ③ 输入文件的名称为'3D scenes'。
- ③ 单击'Save'按钮保存该工程。

完成后的场景：

- ③ ..\Tutorials\1.9 -3D scenes\3D scenes - Complete.cgr

1.10 发布

在某些情况下，一个完成的 Quest3D 工程可以被分发。不管它是一个免费的屏保程序还是一个大型的商业培训程序，Quest3D 的程序必须被导出并保存为一种目标用户能够运行的格式。

根据你的授权号的不同，会有一些不同的发布选项。

可发布的类型	限制版	Professional / Enterprise / VR
.q3d 播放器文件	支持	支持
Web 应用程序	支持	支持
屏幕保护	支持	支持
WinAmp 插件	支持	支持
可单独运行的.exe 文件	不支持	支持
安装文件	不支持	支持

使用限制版发布的工程会在启动的时候出现一个附加的 Quest3D 闪屏。

所有的 Quest3D 程序都需要在目标主机上安装 DirectX 9.0 或更高版本。DirectX 是微软公司提供的免费的扩展包，你可以在 http://www.microsoft.com/directx/download/下载到。

播放器文件.q3d

Quest3D 工程可以发布为一个后缀名为'.q3d'的播放器文件。以这种方式发布的 Quest3D 程序能够使用 Quest3D 的播放器文件来浏览，只需要打开'.q3d'文件即可。

为了能够运行.q3d 文件目标计算机必须安装 Quest3D 的播放器。该播放器可以从 http://www.quest3d.com[大约 1.6 MB]上免费下载。

Internet ActiveX: .q3d 和.htm

'.q3d'文件也可以使用支持 ActiveX 控件的浏览器打开。'.q3d'文件必须被连接到一个适当的网页上才能显示出来。该 Web 页包含 Queset3D 的媒体元素，它将自动从 Quest3D.com 网站上下载并安装 Quest3D 的播放器。当然用户可以取消该操作。

可独立运行的程序：.exe

Quest3D 工程可以发布为一个能够独立运行的'.exe'文件。使用这种类型的发布方式只需要安装 DirectX 即可运行。

安装文件：.exe

安装程序是一个'.exe'的压缩文件，该文件可以拷贝展开程序到硬盘上的一个指定目录下。

实例

下面的实例展示各种类型的发布方式。首先是'.q3d'工程文件，然后是'.exe'程序，最后是安装文件和 Web 应用程序。

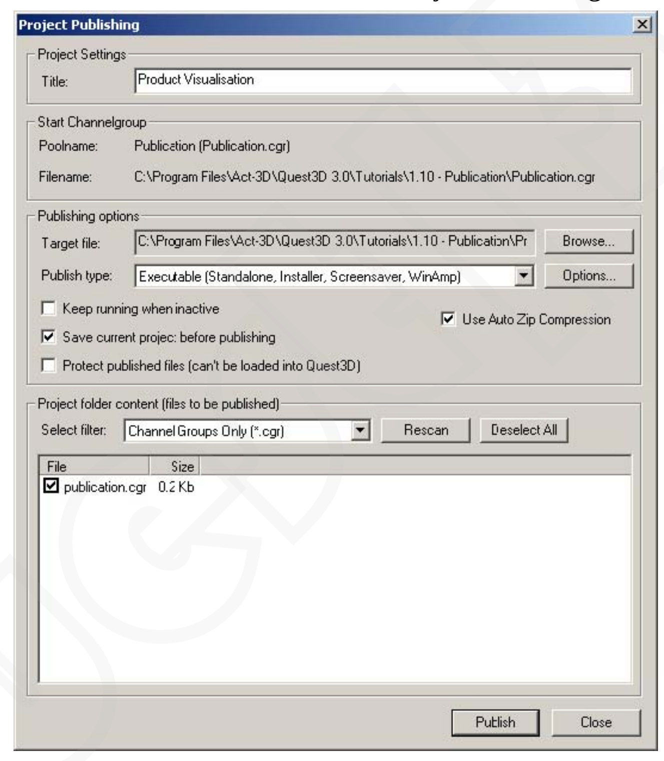
Quest3D 场景：

③ ..\Tutorials\1.10 - Publication\Publication.cgr

Step by step:

③ 打开'Publication.cgr'，该场景包含一个简单的产品展示场景。

③ 从菜单中选择'File > Publish...'。将出现一个'Project Publishing'对话框。



③ 在'Title'栏中输入'Product Visualisation'。

③ 单击'Browse'按钮选择一个需要保存发布文件的目录。

③ 选择'..\Tutorials\2.9 - Publication\'文件夹。

③ 在'File name'栏中输入'Product_Visualisation'。

③ 从'Publish type'下拉列表框中选择'Project File (viewer and web-plugin)'。

③ 钩选'Save current project before publishing'（发布前保存工程）。

③ 在'Project folder content'窗口中确保'Publication.cgr'被选中。

③ 单击'Publish'按钮。

③ Quest3D 询问是否要保存所有的 channel 组。单击'Yes'。这样就可以将 Quest3D 程序发布为.exe 文件并将该文件放置在你的工程目录下。发布时会出现一个进度条，当发布完成后改进度条将消失。

③ 最小化 Quest3D。

- ③ 双击我的电脑图标。

- ③ 找到如下的文件夹：C:\Program Files\Act-3D\Quest3D 3.0\Tutorials\2.9 - Publication\
- ③ 找到名为'Product_Visualisation.q3d'的文件并双击它。这样就可以在 Quest3D 播放器中运行该场景。
- ③ 按住鼠标右键并拖动鼠标就可以绕着三维物体进行环绕。
- ③ 单击'Esc'键退出。

注意为了运行'.q3d'文件，必须安装 Quest3D 播放器和 DirectX 9.0 或者更高版本。

- ③ 返回 Quest3D 编辑程序。
- ③ 'Project Publishing'窗口应该还是打开的，如果没有打开从菜单中选择'File > Publish'
- ③ 从'Publish type'下拉列表中选择'Executable (Standalone, Installer, Screensaver, WinAmp)'。
- ③ 单击'Options'按钮。
- ③ 选择'Self executing project (click and run)'然后单击'OK'按钮。
- ③ 单击'Publish'按钮。该场景将被保存在上述相同的目录下。
- ③ 在目录'..\Tutorials\Publication\'下找到'Product_Visualisation.exe'文件双击该文件即可独立运行该场景。
- ③ 按住鼠标右键并拖动鼠标就可以绕着三维物体进行环绕。
- ③ 单击'Esc'键退出。

注意运行'.exe'文件时不需要启动 Quest3D 播放器。然而必须安装 DirectX 9.0 或更高版本。

- ③ 返回 Quest3D 编辑程序。
- ③ 'Project Publishing'窗口应该还是打开的，如果没有打开从菜单中选择'File > Publish'
- ③ 从'Publish type'下拉列表中选择'Executable (Standalone, Installer, Screensaver, WinAmp)'。
- ③ 单击'Options'按钮。
- ③ 在'Executable Options'窗口中，选择'Project Installer (added to start menu)'。
- ③ 改变'Default installation folder'为'C:\Program Files\Product Visualisation。
- ③ 改变'Start-menu path (folder / title)'为'Quest3D Product Visualisation\Product Visualisation'。
- ③ 钩选'Run project after installation is completed'。
- ③ 单击'OK'按钮。
- ③ 单击'Browse'按钮。
- ③ 改变'File name'为'Product_Visualisation_Installer'并单击'Save'按钮。
- ③ 单击'Publish'。该场景将被保存在与上述相同的目录下。
- ③ 在'..\Tutorials\Publication\'目录下找到'Product_Visualisation_Installer.exe'文件然后双击它。
- ③ 按照屏幕提示将程序安装到你的计算机上。安装完成后，该场景将自动运行
- ③ 测试该场景。

③ 单击'Esc'键退出。
③ 返回 Quest3D 编辑程序。
③ 'Project Publishing'窗口应该还是打开的，如果没有打开从菜单中选择'File > Publish'
③ 从'Publish type'下拉列表中选择'Executable (Standalone, Installer, Screensaver, WinAmp)'。
③ 单击'Options'按钮。
③ 钩选'Create HTML file'。
③ 改变'Title'为'Product Visualisation'。
③ 改变'Filename'为'index.htm'。
③ 单击'OK'确认改变。
③ 单击'Browse'按钮。
③ 改变'File name'为'Product_Visualisation'并单击'Save'按钮以确认。
③ 单击'Publish'。该场景将被保存在与上述相同的目录下。
③ 在'..\Tutorials\Publication\'目录下找到'index.htm'文件双击打开该文件。这样一个 Web 页将在标准的浏览器窗口中打开，Quest3D 的程序将被嵌入到网页中。

> 注意如果要运行 Quest3D 的 Web 应用程序需要安装 Quest3D 的浏览器插件和 DirectX 9.0 或更高版本。如果没有安装浏览器插件，该 Web 页将自动下载并安装它。当然用户可以取消该操作。

实际的应用程序是一个'.q3d'文件。因此'.htm'文件和'.q3d'必须都被上传到 Web 服务器上才能正确地浏览该场景。

完成后的场景

③ ..\Tutorials\1.10 - Publication\Product Visualisation.q3d
③ ..\Tutorials\1.10 - Publication\Product Visualisation.exe
③ ..\Tutorials\1.10 - Publication\Product Visualisation.scr
③ ..\Tutorials\1.10 - Publication\index.htm



1.11 小结

该手册的第一部分介绍 Quest3D 的基本概念。下面列出了该部分的关键点。

③ 创建一个完善的三维场景并保证你的工程具有可管理性的第一步：能够实际的描述你想建什么。Quest3D 将帮助你可视化你的想法。

③ Quest3D 程序是由一些构建模块组成的。这些构建模块被称为'channels'，每一个 channel 都具有特定的功能。

③ 在 channel 的上部或下部的黑色小方块称为’连接块’。不同的 Channel 可以通过连接块相互连接。

③ 具有淡蓝色背景的区域被称为'Channel 视图'。Channel 视图显示的是 Quest3D 程序中所使用所有 channel。

③ 通过从模版列表中拖动一个 channel 到 Channel 视图中可以添加一个新的 channel。

③ 一个’模版’是预定义的一个 channel 或 channel 组，使用模版可以提高效率。

③ 所有的 Quest3D 程序都使用相同的规则。三维物体在程序中起着非常重要的作用。就像一个电影一样，一个 Quest3D 场景需要一个相机和若干灯光。

③ 可以截获用户的输入例如键盘，鼠标，手柄，手写板和虚拟现实设备。

③ 发布一个 Quest3D 工程意味着，将它从编辑器中分离出来并创建一个可独立运行的程序。

第二部分 虚拟场景

摘要

2.1：三维物体

三维物体是虚拟场景的核心。

2.2：动画

不论是随着时间还是基于输入，动画意味着改变。

2.3：导入物体

使用其它软件创建的物体可以被导入和使用。

2.4：面属性

颜色、贴图和透明度所有定义 3 维模型外观的属性。

2.5：光照和阴影

创建一个完美的场景时光照是不可缺少的，该部分介绍了光照的详细信息。

2.6：像机

在用户体验场景的过程中视点起着非常重要的作用。

2.7：图形用户接口

按钮，和滑动条使得用户能够与程序交互。

2.8：声音

模版是预定义的 channel 或 channel 组。这里给出了 Quest3D 功能性的列表。

2.9: 环境

虚拟场景包含许多元素，例如三维物体，像机和灯光。

2.10: 粒子系统

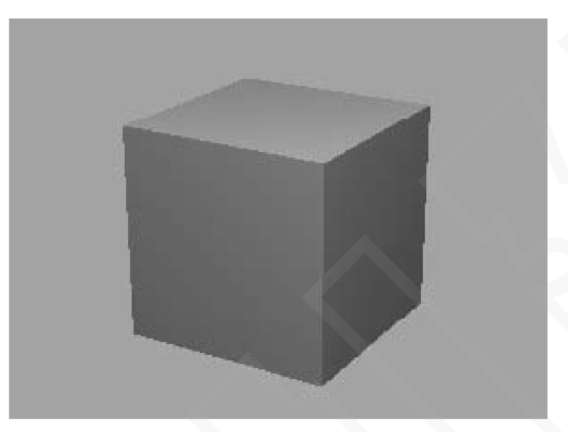
编辑器中制作的工程可以被保存为可单独运行的程序。

2.11: 角色动画

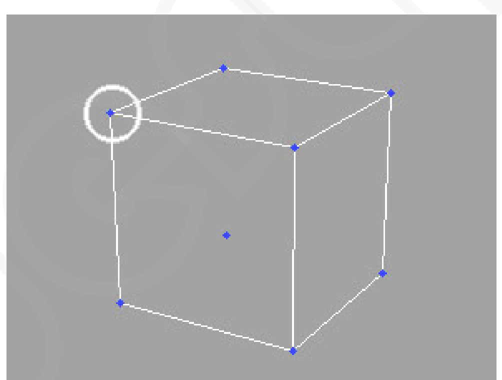
回顾本手册第一部分介绍的内容。

2.1 三维物体

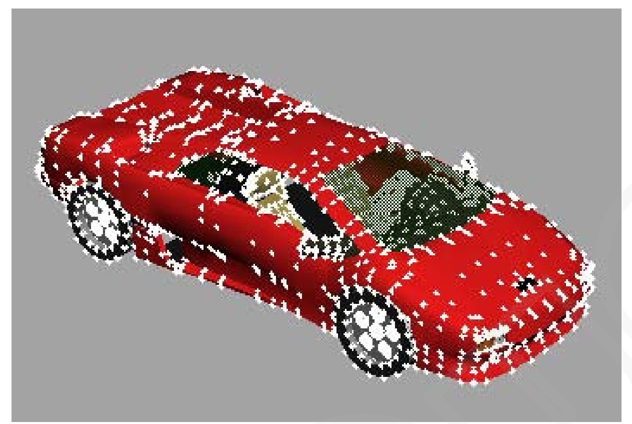
三维场景中会包含一些虚拟物体。这些物体的外观是由许多元素构成的。下图显示了一个立方体。



‘Vertex’ 是位于三维空间中的一个孤立的点。顶点由 X，Y，Z 坐标定义。下图显示了一个 Box 物体的顶点。

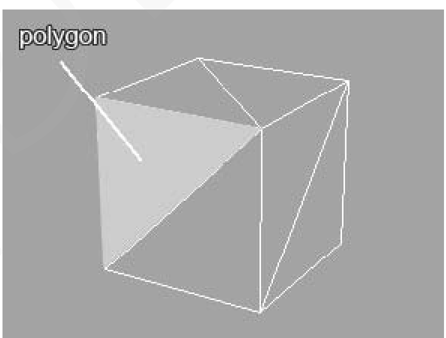


一些模型例如 Box 仅仅由很少的点组成，而一个复杂的物体例如汽车，建筑和角色可能是由数以千计的点组成的。



‘Surface’由’**Polygon’**扩展而来。一个多边形可以有任意多个点，然而在实时图像处理

中，一个多边形通常用三角形和四边形来表示（分别有 3 个和 4 个点）。Quest3D 多边形由三个点组成。下面的图像显示了一个立方体被分成了三角形。

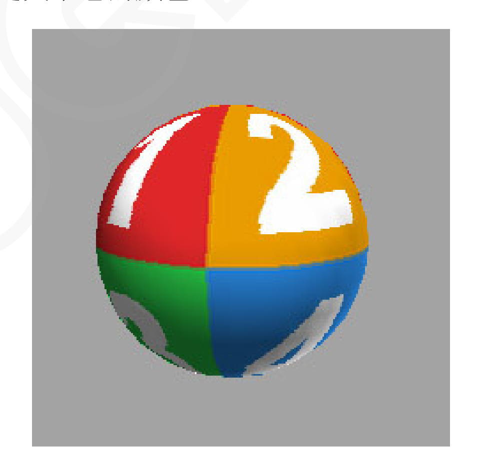


贴图

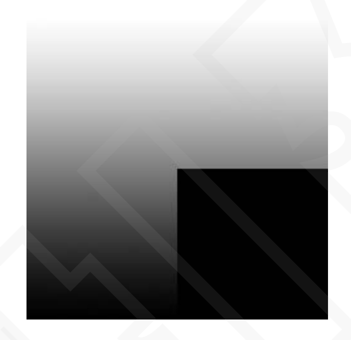
一个多边形可以被赋予一个颜色，但是，通常使用另外的方法来添加细节。'贴图'是可以包裹在三维物体上的图像。如下图所示。



在实时图形处理中，贴图常用来定义**颜色(Color)**和**透明度(Transparency)**。在下图中，贴图包裹在一个球体上并定义了它的颜色。



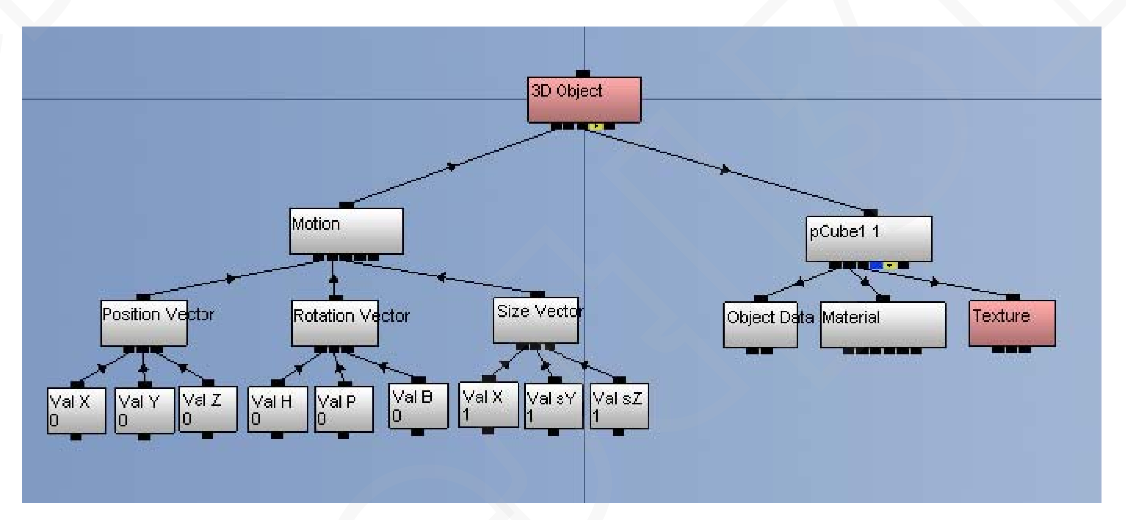
下图显示了一个透明贴图的例子。透明度可以使用从黑（完全透明）到白（完全不透明）的灰度图来定义




在上述球体上使用透明贴图的结果如右图所示。

Quest3D 中的物体

Quest3D 中的物体被划分成两大部分：**Motion** 和 **Surface**。



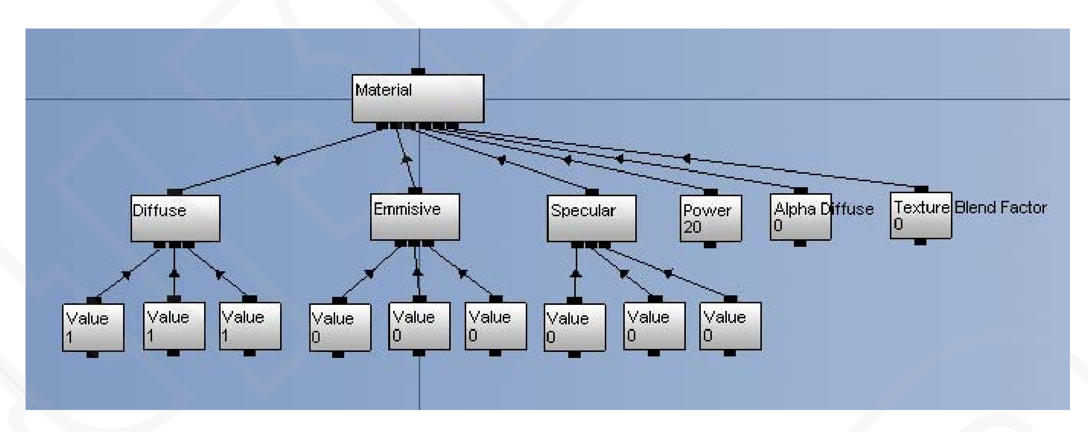
三个 *Vector* 连接到'Motion'channel。他们分别描述了物体的**位置**（**Position**），**旋转**（**Rotation**）和**大小**（**Size**）。

每一个 *Vector* 包含三个 *Value*。如果用于动画制作，这些 *Value* 可以用 *Envelope* channel 来替换。动画将在 2.2 讨论。

Quest3D 中三维模型的实体将出现在 Channel 结构的后半段。按照顺序连接到 *Surface* channel 的分别是 *3D ObjectData（三维物体）, Material(材质)*和 *Texture（贴图）* channel。

3D ObjectData channel 包含一个物体的顶点和多边形，以及如何将贴图包裹在物体上的信息。

Material(材质) channel 被细分为多个部分：Diffuse(散射), Emissive(发射), Specular(反射), Power(强度), Alpha Diffuse(散射)和 Texture Blend Factor（纹理混合因子）.



*Diffuse(散射)*包含物体的颜色，由三个分量组成：红，绿，兰，或’RGB’。

*Emissive(发射)*描述了模型的自发光程度。在没有光的场景中 *Emissive* 为(0, 0, 0)的物体为黑色。

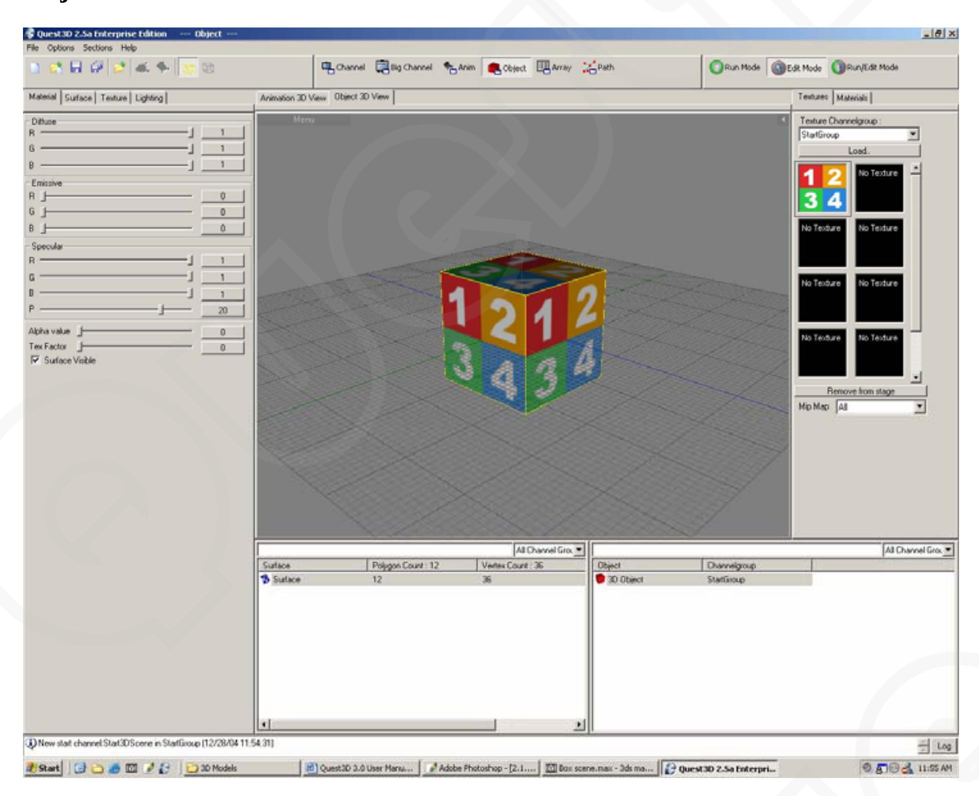
*Specular(反射)*定义了高亮区的颜色和强度，*Power(强度)*的值控制高亮区的大小。*Alpha Diffuse* 定义物体的透明度。

Texture Blend Factor 定义了贴图与散射颜色的混合程度。

Texture channel 代表一个表面的实际贴图。它包含一个散射（颜色）图，还可能包含一个 alpha(透明)贴图。

材质和贴图的设置可以在 Object Section 中调整。

Object Section 的布局



1	菜单	包含标准的功能例如文件管理，定义和帮助
2	工具栏	文件管理的图标，Channel 视图和像机视图
3	Section 选择按钮	在 Section 中快速切换的按钮

4	Run, Run/Edit, Edit 按钮	在三种模式间切换
5	‘Material(材质)’ 标签	物体的材质属性
6	‘Surfaces’ 标签	透明度，混合和物体分类。
7	‘Texture’ 标签	材质贴图，高级透明和混合
8	‘Lighting’ 标签	光照贴图烘焙
9	‘Animation 3D View’ 标签	显示整个场景
10	‘Object 3D View’ 标签	显示选中物体
11	‘Textures’ 标签	列出工程中的所有贴图
12	Surface 列表	列出选中物体的所有面
13	Object 列表	列出该工程中的所有物体
14	日志栏	调试和错误信息

实例

下面的实例集中讨论 Quest3D 中的物体。同时介绍 Object Section

Quest3D 场景

③ ..\Tutorials\2.1 – 3D objects\3D objects.cgr

Step by step:

③ 打开场景‘3D objects.cgr’

③ 单击工具栏上的进入 Object Section。

③ 三维物体显示在 Animation 3D View 窗口中。

③ 单击右下角 Object 列表中名为’Car’的物体以选中它。

③ 在左下角的 Surface 列表中单击’Body’选中它。

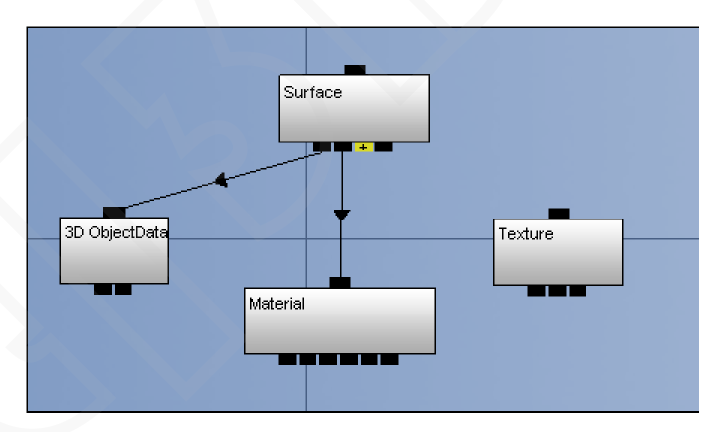
③ 查看左边的‘Material(材质)’ 标签。其中显示了一些滑动条。

③ 拖动‘Diffuse(散射)’到不同的位置，注意该过程中物体颜色的变化。

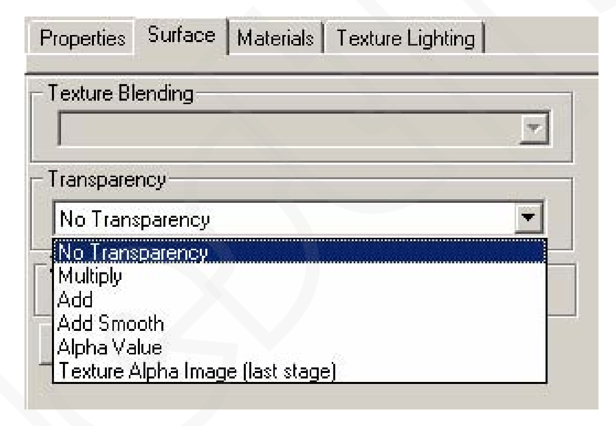
③ 按住‘shift’键然后拖动其中一个滑动条。注意三个滑动条在同时变化。

③ 进入 Channel Section。

③ 将‘Texture’ channel 连接到‘Surface’ channel



③ 进入 Object Section。注意现在贴图显示在模型上
③ 拖动 Emissive 滑动条到不同的位置。注意物体的自发光的变化，该变化与光源无关。
③ 拖动 Specular 滑动条到不同的位置。注意模型上高亮部分的颜色和强度的变化。Power 控制高亮区的大小。
③ 在 Surface 列表中单击’Windows’以选中它。
③ 在使用透明特性之前必须启用它。进入’ Surface’标签。
③ 从‘Transparency’下拉列表中选择‘Alpha Value'注意此时物体消失了。



③ 回到‘Material(材质)’ 标签中。
③ 拖动‘Tex Factor’滑动条到不同的位置。注意物体的透明度是如何改变的。



完成后的场景：

③ ..\Tutorials\2.1 – 3D objects\3D objects – Complete.cgr

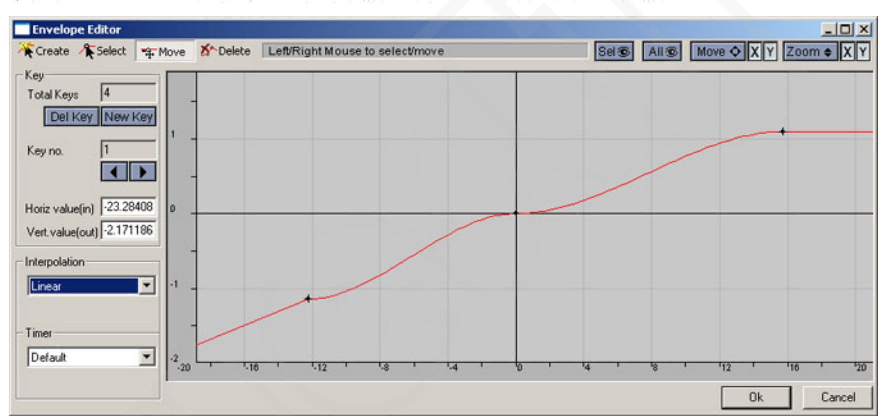
2.2 动画

Quest3D 的强大的特性之一就是它可以使你场景中的任何部分成为动态的。例如，三维物体可以移动，旋转，缩放。此外，所有表面属性都可以成为动态的，例如颜色和透明度。三维角色模型能够使用虚拟骨骼使得它们看起来栩栩如生。可以使用基于位置和时间的触发事件来触发声音。菜单可以实现渐隐效果。

Envelope（封装）

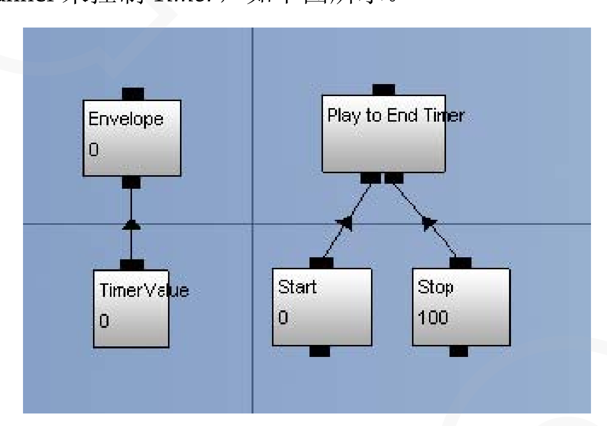
动画意味着某些值随着时间而改变。普通的 *Value* channel 只能存储一个值。为了存储动态数据就要使用 *Envelope* channel。

Envelope channel 的界面类似于一个数学图。水平轴代表输入（通常是时间），可以将一个（*Timer*）*Value* 连接到该 *Envelope* channel 的子连接上实现数值的输入。垂直轴代表输出，通过使用（X，Y）和插值，任何输入都可以转化为一个输出。

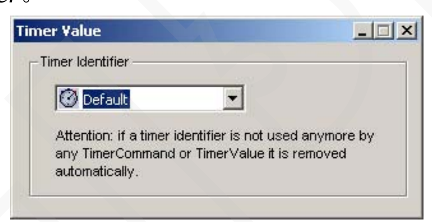


Timers（计时器）

使用 *Timer Value* 和 *Timer Command* channel 可以非常容易的控制动画，*Timer Command*（计时器命令）包含‘Play’，‘Stop’，‘Rewind’和‘Play & Loop’。在下图中，一个 *Timer Value* channel 连接到一个 *Envelope* channel 以作为后者的输入。同时可以使用一个 *Timer Command* channel 来控制 *Timer*，如下图所示。

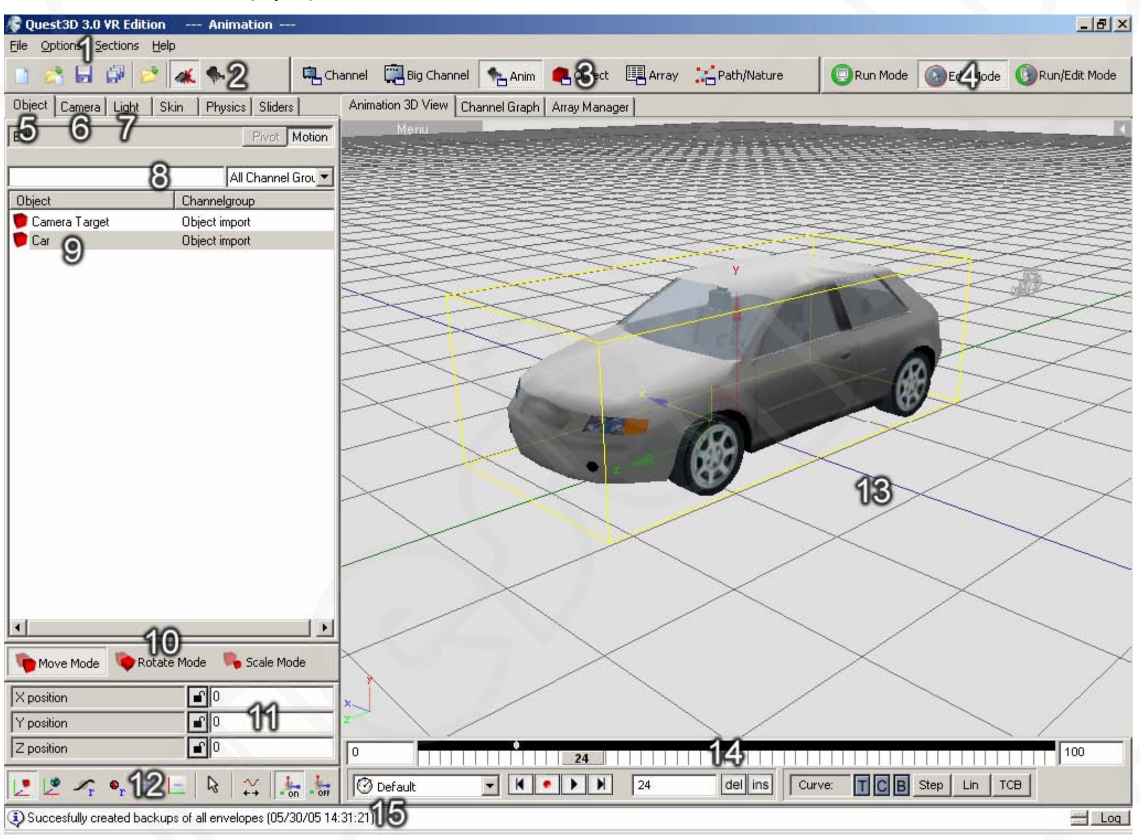


一个 *Timer* 可以在它的属性窗口中命名，在一个场景中你可以根据需要放置多个不同的 *Timer*。



Timer Value channel 会根据计算机的速度自动调整。在 Quest3D 中，25 帧正好为 1 秒。所有的物体均可在 Animation Section 中定位和移动。

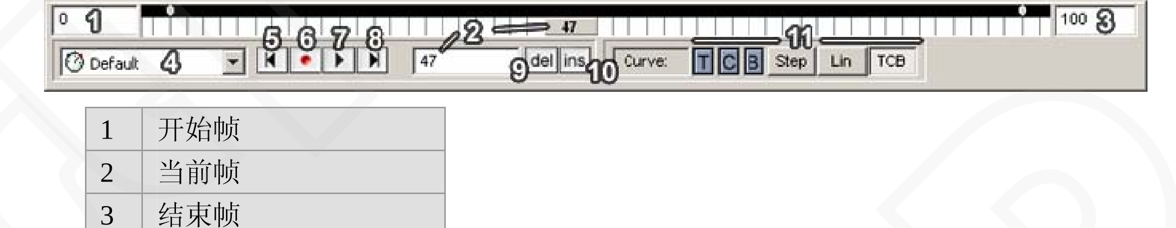
Animation Section 的布局



1	菜单	包含标准的功能例如文件管理，定义和帮助
2	工具栏	文件管理的图标，Channel 视图和像机视图
3	Section 选择按钮	在 Section 中快速切换的按钮
4	Run, Run/Edit, Edit 按钮	在三种模式间切换
5	‘Object’ 标签	显示所有物体的列表
6	‘Camera’ 标签	显示所有像机的列表
7	‘Light’ 标签	显示所有光源和属性的列表
8	搜索栏	在列表中搜索物体
9	Object 列表	列出工程中所有的物体
10	动画控制按钮	移动，旋转和缩放一个物体
11	变换输入栏	在该处输入移动，旋转和缩放的数值
12	配置工具栏	一些控制元素例如世界的缩放

13	Animation 3D View 窗口	显示三维场景
14	时间滑动条	关键帧的创建和每一个 Timer 的动画回放
15	日志栏	调试和错误信息

时间滑动条允许你插入一个关键帧和动画回放。



1	开始帧
2	当前帧
3	结束帧
4	Timer 列表
5	前一个关键帧
6	记录
7	播放(并循环)
8	下一个关键帧
9	删除关键帧按钮
10	插入关键帧按钮
11	Envelope（封装）控制

其他类型的动画

可以使用任何 Basetype 为 *Value* 的 channel 作为动画的输入。例如鼠标移动到物体上时物体的高亮显示，以及通过一个复杂计算控制状态条的长度甚至是智能角色相互之间的行为。

实例

该实例介绍基于时间的 Quest3D 动画。完成该实例后，你应该能够理解 Envelopes, Timers and Timer Commands

Quest3D 场景：

③ ..\Tutorials\2.2 – Animation\Animation.cgr

需要的模版：

③ Animation \ Timer Value (x2)

③ Animation \ Timer Commands \ Play & Loop

③ Animation \ Timer Commands \ Stop

Step by step:

③ 打开‘Animation.cgr’。包含一个简单的场景和两个球体

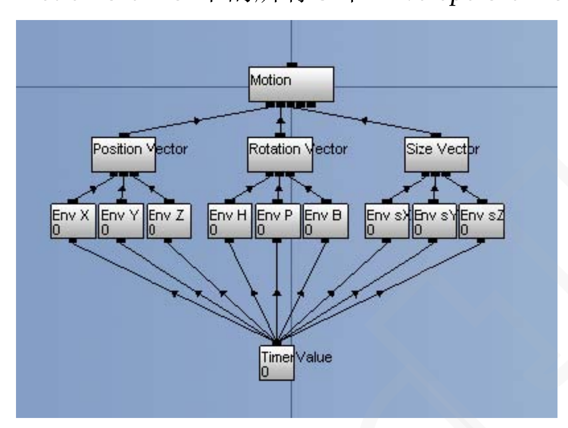
③ 添加一个 *Timer Value* channel 到 Channel 视图中，并将它连接到‘Red Sphere Timer Value’ channel

③ 单击 Anim 按钮进入 Animation Section

③ 单击播放按钮▶。注意此时红色的球开始移动。该动画是循环的。单击停止按钮■。注意动画停止在当前帧

③ 进入 Channels Section

③ 再次添加一个 *Timer Value* 到 Channel 视图中，并将它连接到‘Blue Sphere’物体 *Motion* channel 下的所有 9 个 *Envelope* channel 上



③ 双击 *Timer Value* channel 打开它的属性窗口。从下拉列表中选择‘New..’。再弹出窗口中输入名称‘Blue Sphere’然后单击 OK

③ 进入 Animation Section

③ 在屏幕左边的列表中单击‘Blue Sphere’物体以选择它

③ 时间滑块当前显示的是‘Default’计时器。从 Timer 列表中选择‘Blue Sphere’

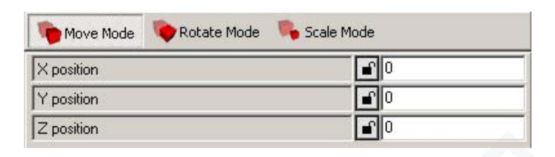
③ 单击播放按钮。注意蓝色的球开始移动，但是红色的球没有移动，这是由于红色球的移动是由另一个定时器控制的（在这里是‘Default’的定时器）

③ 单击停止按钮

③ 在结束帧栏中输入‘250’，注意现在时间滑块被重新调整大小以显示从‘0’到‘250’的所有帧。

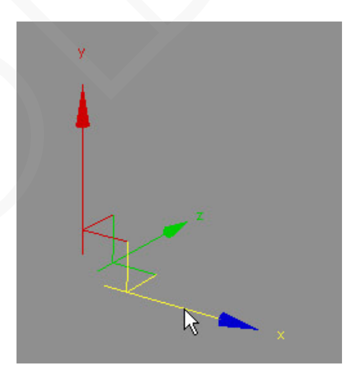
③ 拖动时间滑块到‘150’，然后单击插入关键帧按钮 ins。在弹出的窗口中单击‘OK’以添加一个关键帧。

③ 通过修改左下角变换输入栏中一个或多个坐标的值来移动蓝色的球。

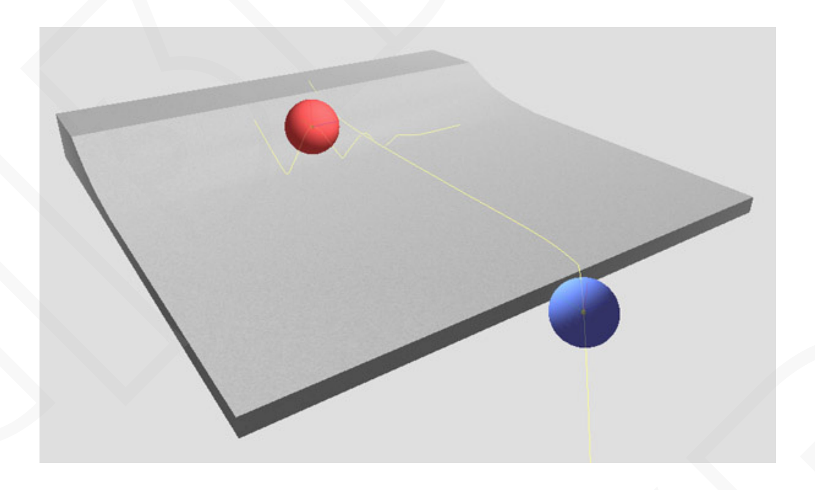


③ 确认 Gizmo 是打开的，该按钮 on 位于屏幕左下角的工具栏上。

③ 通过拖动一个坐标轴来移动蓝色的球。



- 移动时间滑块到第 0 帧，并单击插入关键帧按钮。在弹出菜单中输入 250，并单击 OK。这样第 0 帧物体的状态（位置，旋转和缩放）被拷贝到第 250 帧。现在该动画就能够正确地循环了。
- 单击播放即可查看你刚刚创建的一个比较长的动画。
- 如果需要，做一些必要的调整。
- 进入 Channel Section
- 找到‘Env X’ channel 它是‘Blue Sphere’物体 position vector（位置向量）的一部分。双击‘Env X’ channel 打开它的属性对话框
- 单击关键帧中的一个选择它，并上下拖动以改变它的值
- 使用鼠标右键左右拖动关键帧，这将在水平轴（时间）上改变关键帧的位置
- 可以在在左侧的‘Vert. Value (out)’中输入一个精确的值
- 单击’OK’应用当前的改变并关闭 *Envelope* 编辑窗口
- 进入 Animation Section
- 单击播放按钮查看调整后的动画，然后单击停止
- 进入 Channel Section
- 添加一个 *Play & Loop* 模版到 Channel 视图中。并将它连接到‘Play Animation’ channel.
- 双击‘Play & Loop’ channel 打开它的属性对话框，选择‘Blue Sphere’并单击 OK
- 添加一个 *Stop* 模版到 Channel 视图中。并将它连接到‘Stop Animation’ channel
- 双击‘Stop’ channel 打开它的属性对话框。选择‘Blue Sphere’并单击 OK
- 进入 Animation Section
- 单击 Run Mode 按钮切换到运行模式（Run Mode）
- 按下空格键，动画开始循环播放
- 按下 Esc 键动画停止



完成后的场景：

③ ..\Tutorials\2.2 – Animation\Animation – Complete.cgr

2.3 导入物体

Quest3D 也有可用的三维模型。但是，你肯定会使用一些你自己的模型。许多公司和网站提供了一些完整的三维模型。当然你也可以使用如 3d Studio Max，Maya 或 Lightwave 创建自己的模型。

.X 文件

在使用你自己的物体之前，你需要将它们转化为微软 DirectX 的‘.X’文件格式。‘.X’文件包含了物体的线框，面，贴图和动画数据，并且能够被直接导入到 Quest3D 中。

> 多种建模工具的‘.X’文件导出插件与 Quest3D 的安装包一起发布。可以在 Window 的开始菜单中选择 程序 > Act-3D > Quest 3D 3.0 > Extras > Exporters > 按照程序的目录和版本选择一个合适的插件安装。
> 确定将插件安装到了建模软件的 plug-ins（插件）子目录

附录 A1 详细描述了从 3d Studio Max 和 Maya 中导出和导入模型的过程，此外还描述了使用 Polytrans 导入导出的过程。

实例

在下面的实例中，阐述了一个简单的物体从建模软件中导出为‘.X’格式的，被导入到 Quest3D 中的过程。

准备:

③ 在你最熟悉的建模软件中，创建一个简单的模型，例如立方体。
③ 从菜单中选择‘File > Export’
③ 选择‘.X’文件保存位置
③ 输入名称并单击‘Save’
采用默认的标准设置即可，单击‘Go’按钮

目录‘..\Resources\3D Models\’中包含了一些‘.X’格式的文件，你可以使用这些模型中的一个来完成这个实例。

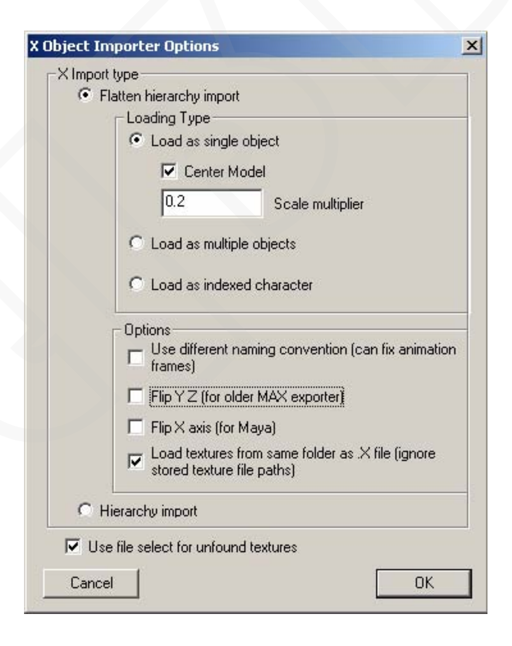
需要的‘.X’ 文件：

③ ..\Resources\3D Models\Cars\Car.X

需要的模版：

③ Scene \ Simple Scene

Step by step:



③ 在 Quest3D 中从菜单上选择‘File > Import’

③ 找到你刚刚创建的‘.X’文件，或者使用上述目录下的‘Car.X’文件。单击‘Open’打开。

③ 弹出的属性对话框提示你输入一个 channel 的名称，改变该名称为‘Car’然后单击‘OK’ 按钮

③ 保持属性对话框中的默认选项不变，单击‘OK’按钮

③ 现在来自于‘.X’文件的三维模型被导入到了 Quest3D 中。该物体存储在标准的 3D Object channel 结构中。

③ 添加一个 *Simple Scene* 模版到 Channel 视图中，将它放置在 object channel 的上方。

③ 选择并删除‘Primitive Object’ channel 和它的所有子 channel。

③ 右键单击‘Start 3D Scene’ channel，在上下文菜单中选择‘Set as start channel’

③ 确定 Animation 3D View 窗口显示当前场景，并使用了投影像机(Project Camera)

③ 将刚刚导入的三维物体连接到‘Render’ channel 上。然后该物体将显示在 Animation 3D View 窗口中。

③ 进入 Object Section

③ 对模型表面做必要的调整。例如‘Emissive(发射)’和‘Specular(反射)’的值以及透明度的设置。

③ 进入 Animation Section

③ 确定 Animation 3D View 窗口显示当前场景，并使用了投影像机(Project Camera)

③ 单击 Run Mode 切换到运行模式（Run Mode）

③ 按住鼠标右键并移动鼠标，视点将绕着物体旋转

③ 可以保存该场景



完成后的场景：

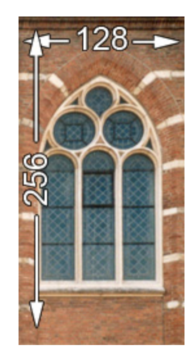
③ ..\Tutorials\2.3 – Object import\Object import – Complete.cgr

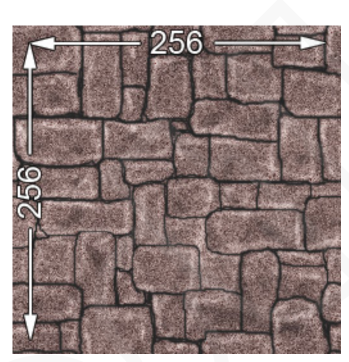
2.4 表面属性

场景中大多数三维模型的外观都可以用它们的表面属性来定义。如贴图，材质贴图，混合方法和透明度。

贴图

贴图是一个可以包裹在三维物体上的图片，在实时图像处理过程中，出于性能的考虑，图片像素大小（宽高）必须是 2 的幂次。在 Quest3D 中，它们必须大于 8。32x32, 128x256 和 256x256 的像素大小都是可用的大小。





图片的像素越大，文件的也越大，并且会增加内存的使用。即使现在的显卡配置了 64M 甚至 256M 的贴图内存，贴图使用不当同样会造成一些问题。一个 2048x2048 像素大小的.jpg 图片会占用至少 40kb 的磁盘空间，还会占用超过 8M 的纹理内存。

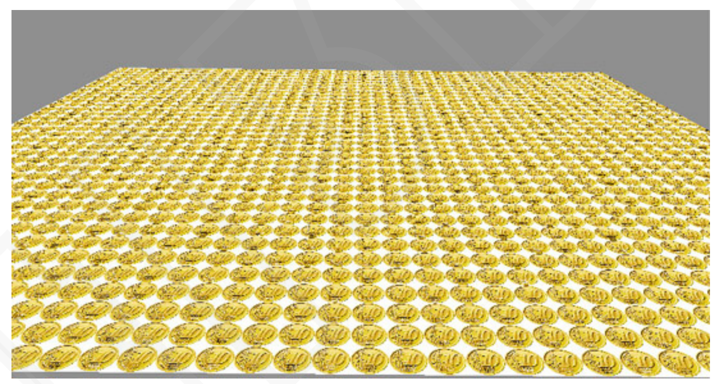
在不牺牲贴图精度的情况下应该使用尽可能小的贴图。一个经验方法是图片的大小不要超过屏幕上显示的大小。例如建立一个远处的场景，128x128 大小的贴图即可。草的贴图不要超过 32x32

压缩

此外，Quest3D 中的贴图可以被压缩。压缩后的贴图质量会有轻微的降低，但是压缩贴图能够减少内存占用大约四倍左右。创建 3D 场景时应该在质量和性能之间做出平衡。

Mipmaps（多纹理映射）

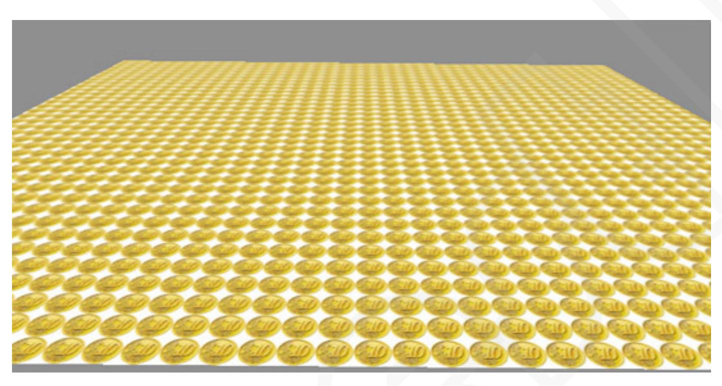
较大的贴图出现在远处物体的表面上时可能会出现失真。



通过使用大小逐渐减小的贴图副本，可以解决这种失真问题。这种技术被称为’mipmaps’可以在 Quest3D 中自动生成这种贴图。




将上面的贴图用在一个平面上并使用 mipmap 的结果如下，



注意较小贴图副本会占用较少的纹理内存。同样，当加载一个场景时，mipmap 必须在启动时生成。而这种处理需要一些时间，所需时间的长短依赖于工程中贴图的数量，大小，指定了 mipmap 的贴图数量。

.dds 图片格式

‘.dds’图片格式可以存储 mipmap 信息，使用这种格式会增加文件的大小，但是 Quest3D 的加载时间会减少，因为在启动的时候不需要生成 mipmap

用于 Photoshop 的‘.dds’插件与 Quest3D 安装文件一起发布。在开始菜单中选择 程序> Act-3D > Quest 3D 3.0 > Extras > Exporters > Photoshop > DDS file 安装时确定将该插件安装到了 Photoshop 的插件目录中。

UV 映射

贴图是按照 UV 坐标被赋在三维模型上的。UV 坐标指定贴图像素放置在模型上的位置。

Quest3D 能够完成简单的 UV 映射方法例如平面（Planar）或立方体（Cubic），如下图所示。这些简单的 UV 映射可以被移动或缩放。

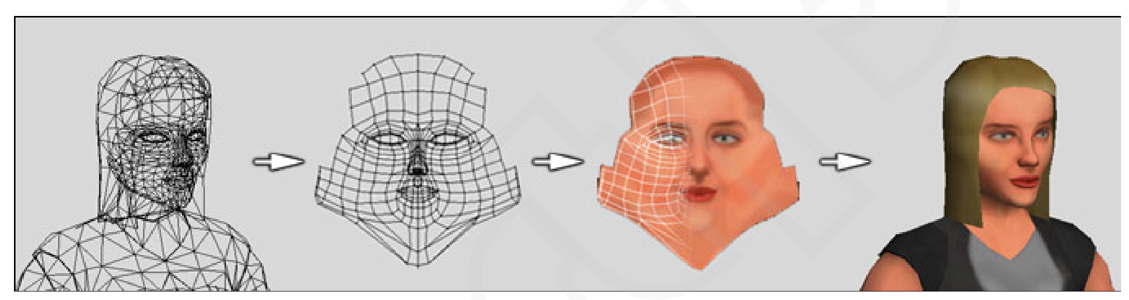


UV 坐标的范围从 0 到 1。(0, 0)表示贴图的左上角，(1.0, 1.0)表示贴图的右下角。



因为 UV 坐标是相对的，物体做了 UV 映射之后，任何大小的贴图均可赋给该物体。

给一个三维模型赋贴图通常需要更加精确的控制。在实际中，UV 映射意味着将一个三维模型的点放置到一个二维图片上。



可以在你最熟悉的三维建模软件中创建复杂的 UV 映射，可以参看建模软件的使用手册获取更多关于 UV 映射的信息。

Quest3D 中 UV 映射的高级形式是‘Reflection map’（反射贴图），这种方法基于当前像机位置将一张贴图投影到一个面上。下图显示了从三个不同角度观察一个使用反射贴图的物体，并给出了所使用的贴图。





Transparency(透明)

可以使用多种方式来处理模型表面的透明度。在 Object Section 的‘Surface’标签中，‘Transparency’下拉列表框提供了许多选项。

Material | Surface | Texture | Lighting
Texture Blending
Multiply
Transparency
No Transparency
No Transparency
Multiply
Add
Add Smooth
Alpha Value
Texture Alpha Image (last stage)

下图中最上面的一行依次显示了使用‘No Transparency’, ‘Multiply’和‘Add’后的不同

效果。下面的一行显示了使用‘Add Smooth’, ‘Alpha Value’和‘Alpha Texture’之后的不同效果。

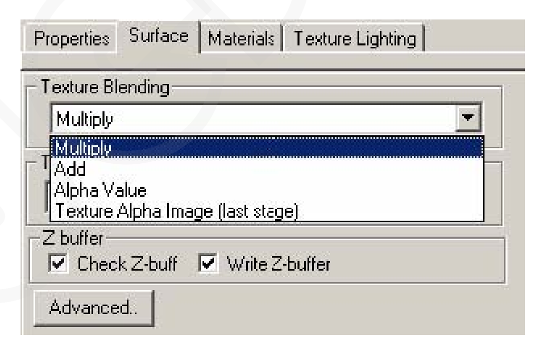


例如‘Multiply’可以产生黑色的窗户或阴影。‘Add’可以产生发光效果。‘Add Smooth’ 可以产生更加精细的发光效果。‘Alpha Value’可以设置一个面的透明度从 0%到 100%。‘Alpha Texture’用于精细的特定透明特效。

贴图 Stage

可以在一个面上使用多个贴图，可以使用的‘texture stages’的数量因显卡而异。大多数显卡支持两个。最新的显卡可以支持三个或四个.

一个面上的不同贴图可以多种方式混合。在 Object Section 的‘Surface’标签中，‘Texture Blending’下拉列表提供了许多选项。这些选项与上面讨论的‘Transparency’下拉列表中的选项基本相同



下图的第一行显示了三个源纹理：stage1, stage2 和 alpha。第二行依次显示了‘Multiply’, ‘Add’, ‘Alpha Value’ (50%) 和 ‘Alpha Texture’。



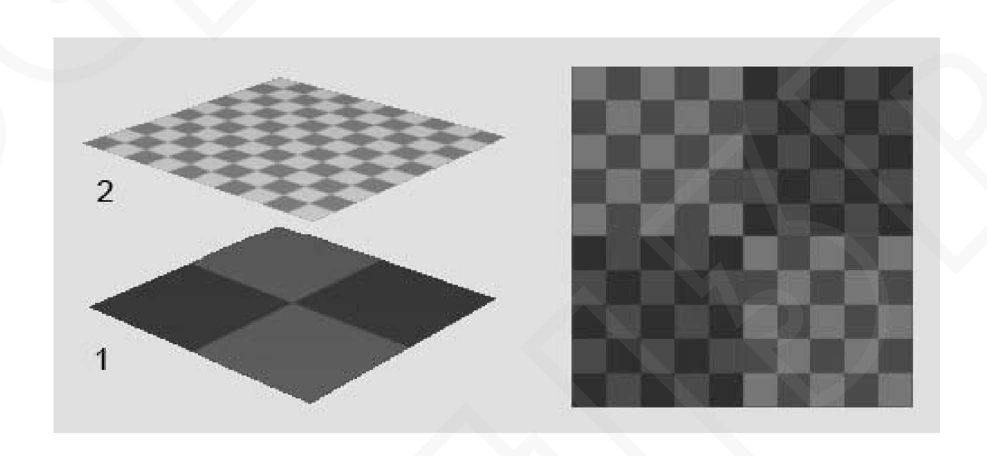
设置‘Multiply’会产生较黑的颜色。‘Add’产生较亮的颜色。‘Alpha Value’混合两个贴图

(0% to 100%). ‘Alpha Texture’用于精确的指定混合。

UV sets

一个面可以拥有多个 UV 坐标。UV 坐标的最大数量因显卡而异，大多数显卡支持两个。用不同的 UV 坐标混合贴图可以产生蒙板叠加（Mask Repetition）效果。某些效果如光照贴图等需要使用其自身的坐标。

下图显示了两个纹理 Stage，每一个都有它自己的 UV 映射。混合后的结果显示在右边。

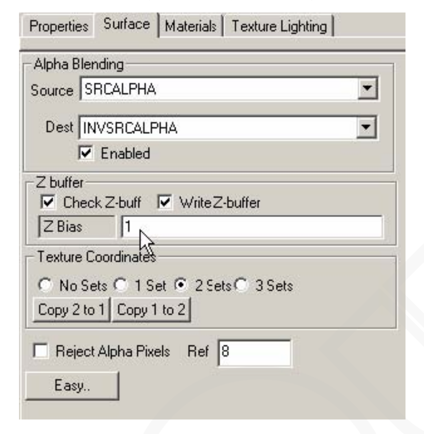


物体顺序

渲染场景时，Quest3D 会自动按照正确的顺序显示所有的面。某些情况下，尤其是存在透明物体的时候，可能会出现面的排序错误。在这种情况下本应该在前一个面（或者面的一部分）会出现在后面或者穿过其他的物体。



为了解决这种问题，通常情况下，透明的面应该在不透明的面之后渲染。另外一种解决方案是设置面的‘Z Bias’。可以在 Object Section 中‘Surface’标签的‘Advanced..’模式下修改该选项。如果没有明确给出面的渲染顺序，一个拥有较高 Z Bias 值的面将首先被渲染。



现在上图中的面被赋予了正确的顺序，显示结果如下所示。

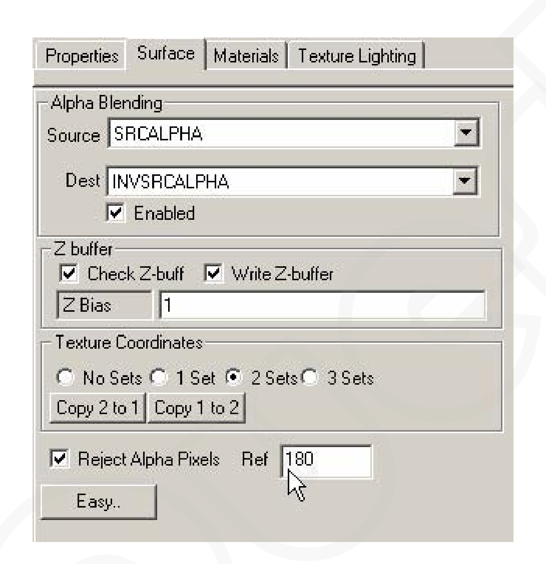


Alpha 失真

用来定义透明度的 Alpha 贴图在某些情况下会出现一些不理想的效果。



需要透明的像素可以通过设置'Reject Alpha Pixels'选项剔除。可以在 Object Section 中'Surface'标签的高级模式中设置该选项。'Ref'的值定义需要剔除的像素的界限。



下图显示了正确设置纹理表面后的效果。



实例

本实例展示了如何使用纹理，透明度，混合方法和面排序。完成该实例后，你应该能够在 Quest3D 中使用纹理。

Quest3D 场景

- ..\Tutorials\2.4 – Surface properties\Surface properties.cgr

需要的纹理

- ..\Resources\Textures\Architecture\Buildings_128.jpg
- ..\Resources\Textures\Landscapes\Grass.jpg
- ..\Resources\Textures\Reflection\Reflection_1.jpg

Step by step:

- 打开场景'Surface properties.cgr'。该场景中包含一条穿过乡村的路，远处有一些建筑，一些树和一个停止的汽车。
- 进入 Object Section
- 单击'Animation 3D View'标签选中它
- 使用投影像机（Project Camera）按钮，确定 Animation 3D View 窗口中显示了这个场景
- 在当前位置上查看远处的建筑，注意到它们相对于场景中的其他物体而言相当小。
- 在列表中单击'Buildings'选中它。
- 在屏幕右边的'Textures'标签中找到名为'Buildings_1024.jpg'的贴图。双击该贴图打开属性对话框。
- 在属性对话框中，注意该贴图的大小和它所使用的内存的大小。
- 单击'Load'按钮并找到'..\Resources\Textures\Buildings\'文件夹。选择文件'Buildings_128.jpg'，然后单击'Open'。新的图片显示在预览窗口中。注意它的大小和使用的内存数量。单击 OK 关闭该对话框。
- 查看远处的建筑，注意现在他们的贴图还是比较精细，在这种情况下出于性能考虑减小贴图的大小并不会导致质量的下降。
- 单击图标切换到编辑像机
- 查看上述场景，注意现在的草地还没有贴图。
- 在列表中单击'Landscape'物体选中它
- 在列表中单击'Grass'面选中它
- 在右侧的'Textures'标签中，单击'Load'按钮加载一个新的贴图。在'..\Resources\Textures\Landscapes\'文件夹中找到文件'Grass.jpg'，并单击'Open'选择它然后单击 OK 关闭该对话框。
- 单击新的贴图以便将其应用到'Landscape'物体的'Grass'面上。注意该贴图被拉伸以贴满整个地形
- 在左侧的'Textures Stages'标签中，从'Mapping'下拉列表中选择'Planar Y'。注意贴

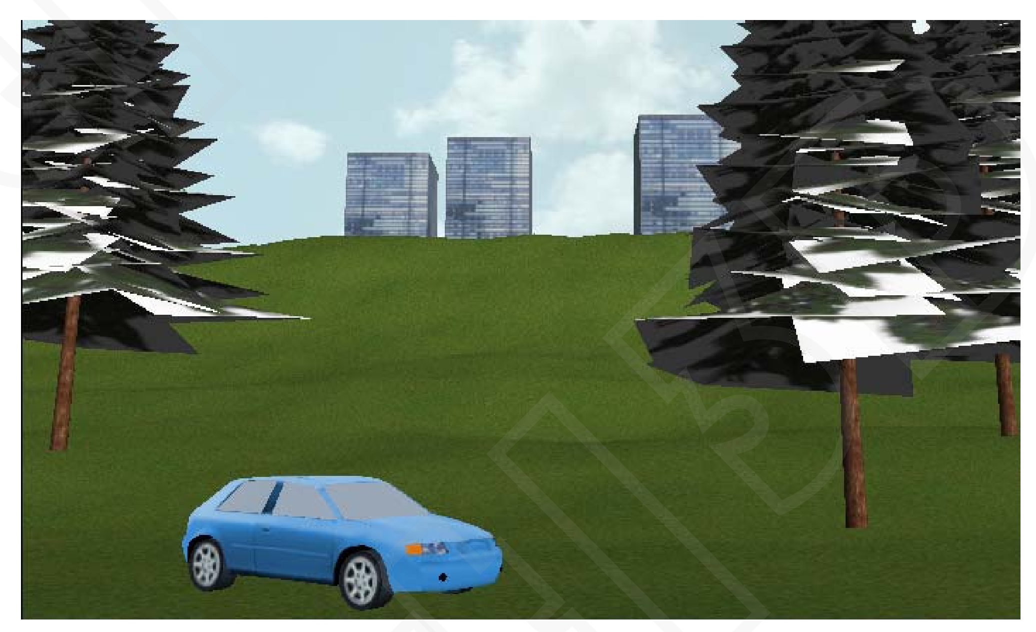
图沿着垂直方向被投影到物体的表面。

③ 拖动‘Scale’按钮来改变贴图平铺。注意该面贴图的变化。

③ 在‘Scale’按钮后面的‘U’和‘V’栏中输入 20 来精确设置平铺。

③ 单击图标使用投影像机（Project Camera）显示场景。

③ 在当前视图中查看草地。注意远处的失真。



③ 在屏幕右侧的‘Textures’标签中，双击名为‘Grass.jpg’的贴图，打开属性对话框，注意 MipMap 的数量值为 1，将该值改变为 0 以便 Quest3D 能够自动决定需要多少级 mipmap。单击 OK 关闭该对话框。

③ 再次查看草地。注意到远处的失真已被修复。

③ 单击以便使用使用编辑像机来显示场景。

③ 在右下角的 Object 列表中，选择名为‘Car’的物体。

③ 在屏幕下方的 Surface 列表中单击名为‘Chassis’的面以选中它。

③ 在右侧的‘Texture Stages’标签中，单击‘Stage 2’标签。

③ 在右侧的‘Textures’标签中，单击‘Load’按钮以加载一个新的贴图。在‘..\Resources\Textures\Reflection\’文件夹中选择‘Reflection.jpg”单击‘Open’打开，然后单击 OK 关闭该对话框。

③ 单击新的贴图以便在‘Sports Car’物体的‘Chassis’面上使用它。注意该贴图被拉伸到整个汽车。

③ 在左侧的‘Texture’标签中，从‘Mapping’下拉列表中选择‘Reflection map (CamReflvec)’。注意贴图的变化

③ 绕着汽车旋转一下注意贴图产生的变化

③ 在‘Surface’标签中，从‘Texture Blending’下拉列表中选择‘Alpha Value’

③ 在‘Material(材质)’标签中，拖动‘Texture factor’滑块到 0.92。注意 reflection stage 与下面的 Diffuse texture stage 混合在一起。

③ 绕着汽车旋转一下注意反射产生的变化。如果有必要，调整‘Texture Factor’滑块来改变混合量。

③ 在列表中单击名为‘Windows’的面。

③ 在‘Surface’标签的‘Transparency’下拉列表中选择‘Multiply’。注意车窗变暗而且透明了。
③ 在‘Transparency’下拉列表中选择‘Add’，注意现在车窗变亮了，几乎在发光。
③ 在‘Transparency’下拉列表中选择‘Alpha Value’。注意到现在车窗变的完全透明了。
③ 在‘Material(材质)’ 标签中拖动‘Alpha’滑块到 0.5。注意车窗现在变为 50%透明了。



③ 查看场景中的树。注意树叶周围的平面应该是透明的。
③ 在列表中单击‘Trees’物体以选择它。
③ 在列表中单击名为‘Leaves’的面选择它。
③ 在‘Surface’标签中的‘Transparency’下拉列表中选择‘Alpha Texture’。注意树叶四周的平面变得透明了一点，但是还有一些不理想的轮廓。
③ 在‘Surface’标签中单击‘Advanced..’按钮
③ 选择‘Reject Alpha Pixels’并输入 150 做为‘Ref’值。注意树叶周围的轮廓消失了



完成后的场景

③ ..\Tutorials\2.4 – Surface properties\Surface properties – Complete.cgr

2.5 光照和阴影

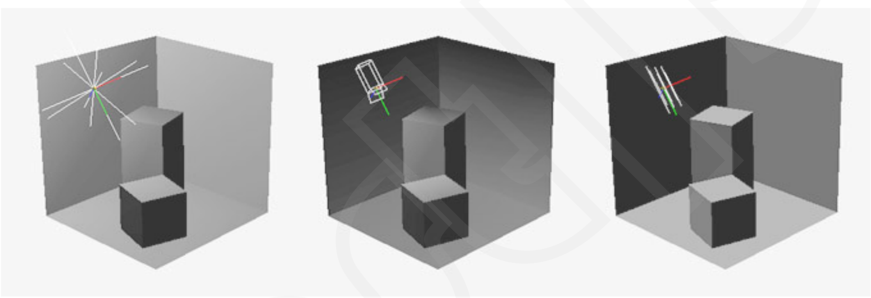
适当的光照对于建立一个完美的场景来说是必须的。光照和阴影能给用户带来视觉上冲击。如果没有阴暗对比，很难在场景中建立景深效果。

可以在 Quest3D 中使用许多光照技术：常用的外部光，自发光，实时阴影和光照贴图。本章详细描述了这些技术。

通常外部光都包含一个能够在场景中发光的源。在 Quest3D 中光照实际上是一个 channel，它还可以有一些子连接。光的属性可以在 Animation Section 的'Light'标签中设置。

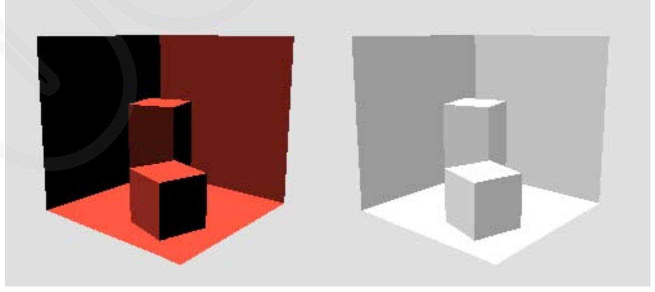
光照类型

Quest3D 中的光照可以有三种类型：点光源，透射光和方向光，如下图所示。



点光源向任何方向发射光。投射光由外角和内角来定义，形成一个渐变的效果。方向光只沿着一个方向发射光线。所有方向光的光线都相互平行。

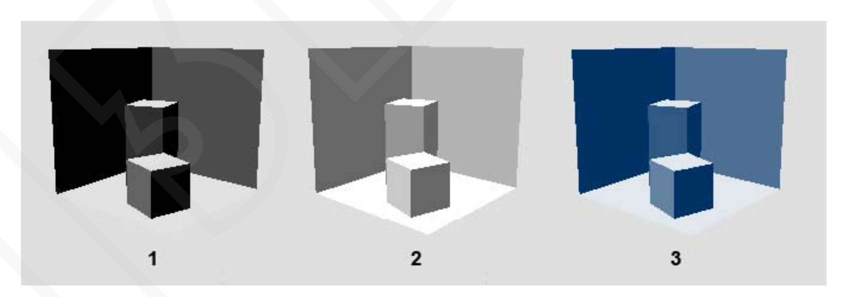
Qeust3D 中光线的颜色可以通过它的'Diffuse(散射)'来设置。'Ambient'向量就像一个全局的'Emissive(发射)'向量，场景中的所有物体均会受到它的影响。



上图中显示了'Diffuse(散射)'和'Ambient'光的例子。

可以通过连接一个 *Light* channel 到 Render 上来控制场景中光照的效果。只有连接到同一个 *Render* 上的物体才会受到该光源的影响。每个 *Render* channel 最大可以连接 8 个光源。该数量受硬件的影响。

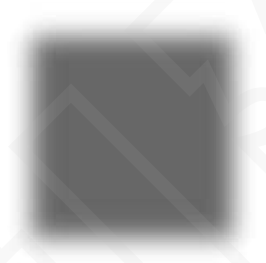
物体的亮度可以通过它的 Diffuse(散射) 和 Emissive(发射)来改变。可以在 Object Section 中的'Material(材质)'标签中设置这些选项。较大的 Emissive(发射)意味着某种程度上的自发光或环境光。尤其在室外，增加 Emissive(发射)将产生非常好的效果。如上所述，光源的'Ambient'向量就像一个全局的'Emissive(发射)'向量能够影响所有物体。



1) 模型 Diffuse(散射) (1, 1, 1)，光照 Diffuse(散射) (1, 1, 1)，光照 Ambient (0, 0, 0)
2) 模型 Diffuse(散射) (1, 1, 1)，光照 Diffuse(散射) (1, 1, 1)，光照 Ambient (0.4, 0.4, 0.4)
3) 模型 Diffuse(散射) (1, 1, 1)，光照 Diffuse(散射) (1, 0.8, 0.6)，光照 Ambient (0, 0.2, 0.4)

阴影

可以使用贴图来模拟动态物体的阴影效果，例如汽车或是走动的角色。



这样的阴影面可以被快速渲染，并能够得到比较好的效果。



实时阴影：

Quest3D 也支持实时阴影。*Stencil Shadow* channel 可以被看作是 Render 在计算阴影方面的加强。*Stencil Shadow* channel 需要一个光源位置的 *Vector* 才能工作。通过 *SoftwareStencilShadowObject* 可以将一个三维物体连接到 *StencilShadow*。

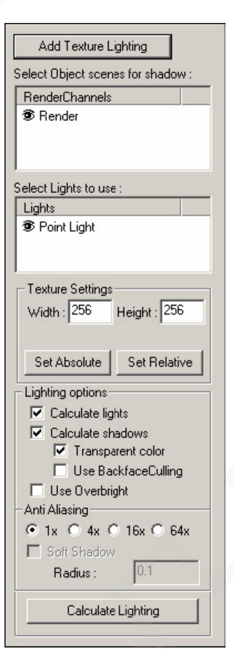


可以使用该 channel 的硬件版：FastShaderStencilShadowObject，该 channel 效率较高但是不支持非闭合物体或者蒙皮。

Lighting Mapping（光照贴图）

实时阴影需要大量的处理。而且，实时阴影不仅粗糙而且不会考虑环境光的影响。如果要创建平滑的而高效的光影，可以使用光照贴图（Lighting Mapping）来实现。





光照贴图是处理一次光照和阴影并将这些信息存储在贴图中的过程。然后光照贴图作为第二个纹理 stage 与 Diffuse(散射) 贴图混合并应用到物体上。

许多三维建模软件都可以生成光照贴图，例如 3D Studio Max 和 Maya，在 Quest3D 中也可以创建一个光照贴图。

在 Object Section 的'Texture Lighting'标签中，有一些可以用来定义光照贴图的设置。可以选择渲染和灯光，也可以设置纹理的宽高。可以计算光照和阴影。还以使用反走样使光照贴图更加平滑。

在计算一个物体的光照贴图之前，它需要有一个 UV。对于一个相对简单的物体如平面或地形而言，可以使用 Quest3D 的 planar UV 映射。当然可以在三维建模软件中调整一个更加精细的 UV 映射。

实例

Quest3D 场景

③ ..\Tutorials\2.5 – Lighting and shadows\Lighting and shadows.cgr

需要的模版

③ Scene \ Lights \ Point Light

Step by step:

- ③ 打开场景‘Lighting and shadows.cgr’。该场景包含一个没有灯光的街区和一个动画角色。
- ③ 添加一个 *Point Light* 模版到 Channel 视图中并将它连接到‘Animated Character Render’ channel。注意角色受到灯光的影响。
- ③ 创建一个‘Point Light’ channel 的快捷方式并将它连接到‘City Render’ channel，注意环境受到灯光的影响。
- ③ 进入 Animation Section
- ③ 在‘Light’标签中选择‘Point Light’并在 Animation 3D View 窗口中移动它，注意角色和环境受到灯光的影响。
- ③ 移动光源到位置(0, 0, 5)
- ③ 改变旋转角为(0, -90, 0)。注意旋转一个点光源并不会改变它的影响。
- ③ 单击‘Spot’按钮改变‘Light Type’。注意光效的改变。
- ③ 改变‘Outer Cone Angle’为 120，‘Inner Cone Angle’为 30。注意着两个角之间的过渡光。
- ③ 单击‘Dir’ (directional)按钮改变‘Light Type’。注意光效的变化。
- ③ 旋转光源到(30, -45, 0).
- ③ 移动光源到(-100, 30, 100)。注意移动方向光并不影响它的效果。
- ③ 进入 Channel Section
- ③ 改变‘Point Light’ channel 的名称为‘Sun Light’
- ③ 添加一个 *Vector* channel 并将它连接到‘Sun Light’ channel.的‘Ambient’子连接上。改变它的值为(0.4, 0.4, 0.4)。注意角色和环境都变亮了，就像受到了日光的影响。
- ③ 添加一个 *Render Stencil Shadow* channel 到 Channel 视图中并将它连接到‘Project’ channel。
- ③ 创建‘Sun Light’ channel 的位置向量的快捷方式并将它连接到‘Shadow Renderer’ channel 的‘Light vector’子连接上。
- ③ 添加一个 *Software Stencil Shadow* Object channel 并将它连接到‘Render Stencil Shadow’ channel 的‘Stencil Shadow Base Object’子连接上。
- ③ 创建‘Animated Character’ channel 的快捷方式并将它连接到‘Software Stencil Shadow Object’ channel。注意现在这个角色在环境中已经出现了阴影。
- ③ 进入 Animation Section。
- ③ 移动‘Animated Character’，注意角色的影子从太阳的方向投射过来。
- ③ 进入 Object Section
- ③ 在列表中单击‘Street’物体
- ③ 在列表中单击名为'Street 1'的面选择它
- ③ 在‘Texture’标签中单击‘Stage 2’标签。从下拉列表中选择‘UV Set 1’
- ③ 在‘Lighting’标签中单击‘Add Texture Lighting’按钮。注意一个黑色背景的绿色网格出现在‘Street’模型上
- ③ 选择‘Environment Render’但不要选择‘Animated Character Render’，眼睛上面有红色叉的符号代表该 render 没有选中
- ③ 选中‘Sun Light’
- ③ 设置纹理的‘Width’和 ‘Height’分别为 256。单击‘Set Absolute’按钮。注意带有黑色背景的绿色网格更新了
- ③ 确定‘Calculate lights’和‘Calculate shadows’选项被选中而其他的选项没有选中。

- 设置‘Anti Aliasing’为 1x
- 单击‘Calculate Lighting’按钮以使用当前设置创建一个光照贴图。注意弹出了一个进度条，计算光照贴图需要一定时间。计算完成后，‘Street’物体现在显示了光照阴影贴图
- 在‘Surface’标签中的‘Texture Blending’下拉列表中选择'Alpha value'
- 在'Material(材质)'标签中设置'Tex Factor'为 0.75。注意现在光照贴图与纹理混合了。
- 在‘Lighting’标签中调整‘Width’ 和‘Height’为 1024，并设置‘Anti Aliasing’为‘4x’
- 单击‘Calculate Lighting’按钮以使用当前设置创建最终的高解析光照贴图。注意计算时间增加了
- 等光照贴图计算完成后，查看结果
- 为'Street'物体剩下的面创建光照贴图



完成后的场景：

- ..\Tutorials\2.5 – Lighting and shadows\Lighting and shadows – Complete.cgr

2.6 像机

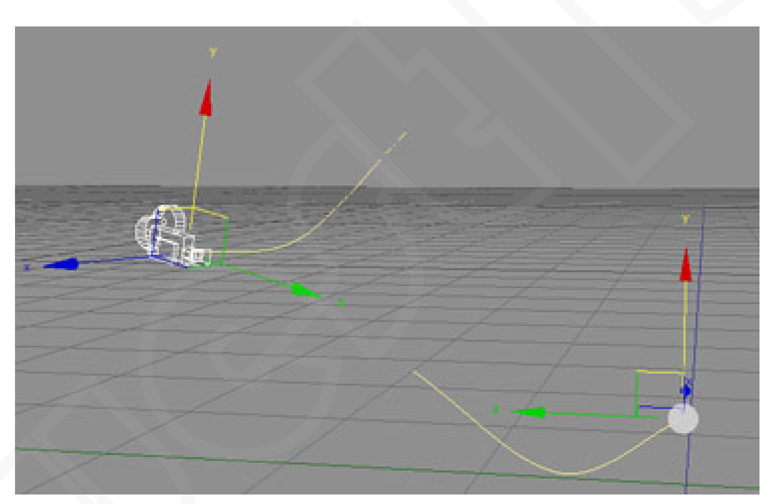
像机定义了从哪个视点来查看场景。为一个工程选择正确的像机是非常重要的，因为它会直接影响到用户体验场景的方式。本章讨论了一些可用的像机模版和常用的选项。

Animation Camera（运动像机）

一个运动像机可以带领用户穿过一个场景，就像现实世界中的手持式摄像机。这种像机是创建非交互式场景的首选，例如展示，记录或屏保。

Animation Camera with Target（带目标的运动像机）

一个带有目标的运动像机总是指向一个不存在的辅助物体。像机和辅助物体都可以移动。



Object Inspect Camera（物体检视像机）

Quest3D 是一个成熟的产品展示制作软件。一个物体检视像机能够绕着产品的三维模型旋转，从而使用户能够从任意角度来观察它。这种产品可能像科技器件一样小或者像整个街区一样大。

1st Person Walkthrough Camera（第一人称行走像机）

通常我们想要的是能够像在现实世界中一样体验一个场景。第一人称行走像机以眼睛的角度查看一个场景，并允许用户自由行走。由于这种方式非常接近现实世界，因此这种类型的像机提供了一种沉浸式的体验。许多计算机游戏都使用了第一人称的视角。

3rd Person Walkthrough Camera（第三人称行走像机）

相对于第一人称视角，第三人称视角是将像机放置在虚拟角色的后面。该像机依附于这个角色或者化身并一直跟随着它。

这种像机具有较少的沉浸感。然而，化身可以看作是一个代理人，通过它用户能够与虚拟世界进行交互。对于某些用户来说第一人称视角过于混乱，并倾向于使用第三人称视角。在某些情况下，第三人称视角提供了更多关于化身环境和位置的信息。

除了视点外还可以定义像机其他方面的属性。

Zoom Factor（变焦因子）

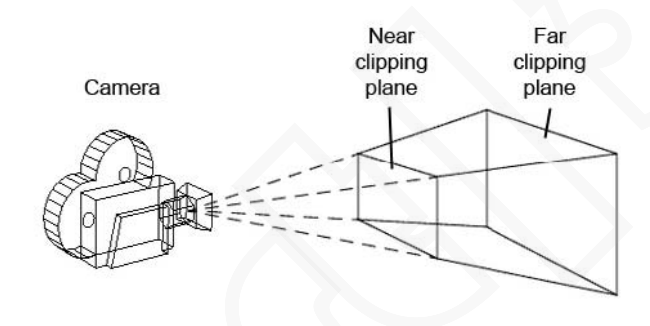
变焦因子是一个快速的缩放像机的方法。缺省的值 1 表示一个标准视图。较大的缩放因子会拉近场景，较小的值会推远场景。

Clipping Planes（裁减平面）

裁减平面是一个像机的虚拟边界，在这个边界之内的物体会被渲染，而边界之外的物体不会被计算。

通过附加一个 *Field of View Matrix* 模版到 *Camera* 上，可以手动定义裁减面，该模版允许改变视野的角度（度）。标准的值是从 50 到 90 度。

一旦裁减面确定了，需要渲染的几何体的数量就会减少，程序的性能因此而提高。



雾

雾用来把虚拟场景中的各种元素混合在一起。可以将它添加为场景的大气来表示景深。使用雾的好处是像机的远裁减面可以缩小到‘Fog End’。如果使用正确，这将会在场景和远裁减面之外的部分之间实现平滑过渡。

实例

Quest3D 场景

③ ..\Tutorials\2.6 – Cameras\Cameras 1.cgr

需要的模版

③ Scene \ Cameras \ Object Inspection Camera

Step by step:

③ 打开场景‘Cameras 1.cgr’。该场景包含一个汽车。

- 拖动一个 *Object Inspection Camera* 模版到 Channel 视图中。将它连接到‘Render’ channel。
- 进入 Animation Section
- 确定 Animation 3D 视窗使用投影像机（Project Camera）显示场景。
- 切换到运行模式（Run Mode）（Run Mode）
- 单击并按住鼠标右键，拖动鼠标绕着汽车旋转
- 切换到编辑模式（Edit Mode）。

完成后的场景：

- ..\Tutorials\2.6 – Cameras\Cameras 1 – Complete.cgr

你可以保存当前 Channel 组到工程目录下。下面的步骤将产生一个新的场景。

Quest3D 场景:

- ..\Tutorials\2.6 – Cameras\Cameras 2.cgr

需要的模版

- Scene \ Camera \ Walkthrough Camera
- Objects \ Collision \ Collision Object

Step by step:

- 打开场景‘Cameras 2.cgr’。进入 Channel Section。该场景包含一个城区和一些建筑，这些物体连接到一个没有像机的 render 上。城市的三个部分‘Car’, ‘Building_Collision_Objects’ 和 ‘Street’将被用做碰撞物体。
- 切换到编辑模式（Edit Mode）。
- 添加一个 *Collision Object*，并将‘Street’ channel 连接到它上面，留下‘Street’ channel
- 双击 *Collision Object* 并点击‘Create Tree’。该物体将被计算
- 拖动一个 *Walkthrough Camera* 到 Channel 视图中。先不要连接到‘Render’ channel
- 将 *Collision Object* 连接到 *Walkthrough* 模版下 *Fast Collision Response* channel 的最后一个子连接上
- 为建筑物创建一个 Collision 物体，将‘Building_Collision_Object’ channel 从‘Render’上断开，只留下‘Car’ channel
- 将‘Walkthrough Camera’ channel 连接到‘Render’ channel。注意 Animation 3D 视窗备耕行了，并且从另一个角度显示场景。
- 将‘Camera Logic’ channel 连接到‘Start3DScene’。如果按下空格，像机将被重置到它的原始位置。
- 在‘In: Collision Spheroid Radius’（ ‘FastCollisionResponse’的第二个子 channel）文件夹上双击。该文件夹包含一个 *Vector*，用来设置像机的高度，改变 Y 值，例如 1.9。
- 进入 Animation Section
- 切换到运行模式（Run Mode）
- 使用方向键和鼠标在场景中行走以测试场景。

③ 单击空格键重置像机。



完成后的场景

③ ..\Tutorials\2.6 – Camera’s\Camera’s 2 – Complete.cgr

2.7 图形用户接口

用户的输入能够影响应用程序。可以和应用程序交互的媒介称为接口。键盘和鼠标是大多是程序所使用的接口。出现在计算机屏幕上的接口称为图形用户接口，或者是 GUI

图形用户接口通常作为一个二维层显示在屏幕的最前端。光标，按纽，滑动条和输入框都是 GUI 的一部分。

Head-up Display

非交互的二维覆盖图是 Head-up Display（HUD）的一部分。例如计数器，公司 LOGO，虚拟场景的顶视图和代表当前用户在游戏中的当前状态。



有时术语‘Graphic User Interface’ 和 Head-up Display’具有相同的含义，在上图中 Quest3D 的 LOGO 和调试信息是 HUD 的一部分而 RGB 值和旋转选项是 GUI 的一部分。

Quest3D 中的 GUI

在 Quest3D 中，屏幕上的二维层使用特殊的方式来渲染。它实际上是另一个三维场景，可以用它自己的像机以一种独特的方式来渲染。如果关闭了反走样 GUI 贴图会变得不平滑。

Z Buffer（Z 缓存）

在每一帧中，首先渲染整个三维场景，然后是 GUI 和 HUD 层。在此期间需要清空 Z 缓存，以防止两者的交叠。

鼠标输入

可以使用 *Mouse Over* channel 和 *User Input* channel 来捕获用户鼠标与 GUI 元素的交互。

二维图像

使用 *Copy image* channel 可以将一个二维像素图渲染到屏幕上。与屏幕分辨率无关，这个图片总是以绝对像素显示在屏幕上。这种方法可以产生非常清晰的图像。

文字

使用 *Text Out* channel 可以将二维文字显示在屏幕上。使用这种方法可以显示帧率和视口分辨率等调试信息。

模版

可以从模版列表中找到许多可用的 GUI 元素的模版。

实例

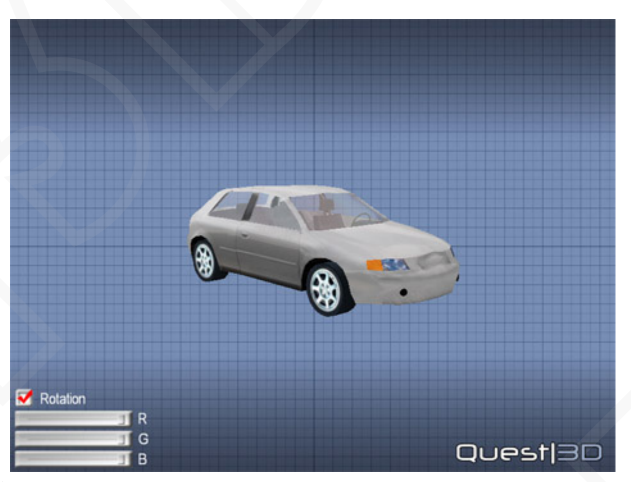
包含按钮和滑块的图形用户接口允许用户改变应用程序。下面的实例展示了这些元素。

Quest3D 场景。

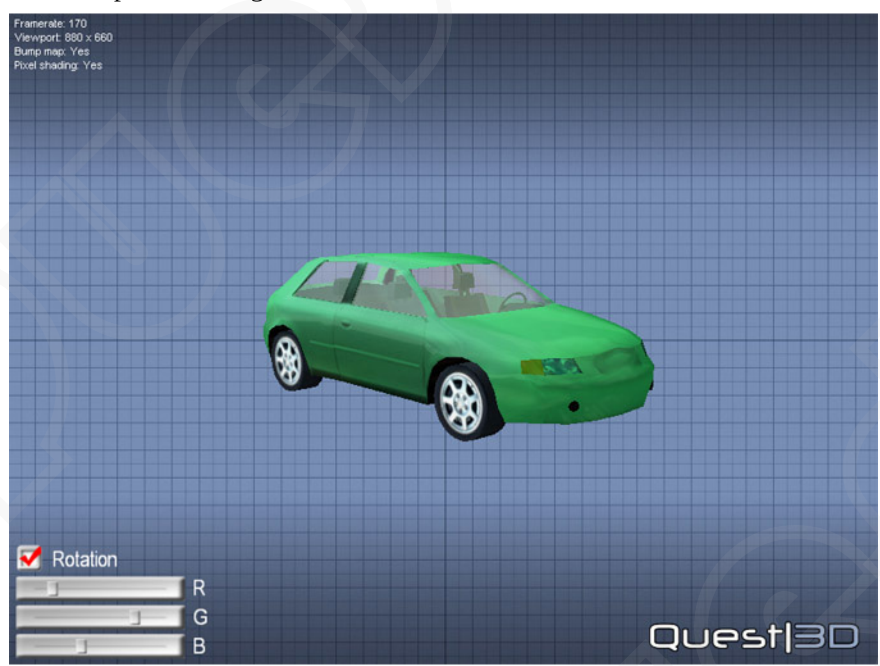
- ..\Tutorials\2.7 – Graphic user interface\Graphic user interface.cgr

Step by step:

- 打开‘Graphic user interface.cgr’场景。该场景包含一个汽车和一个 Object Inspection Camera。注意 Quest3D 的 LOGO 显示在在 Animation 3D View 窗口的最前端。
- 为‘Checkbox’ channel 创建一个快捷方式并将它连接到‘Mouse Over Checkbox’ channel
- 为‘Checkbox State’ channel 创建一个快捷方式并将它连接到’Rotation’ channel，这个 channel 是‘Car’物体 Motion 的一部分。可以在‘3D Scene’文件夹中找到这个‘Car’物体。
- 进入 Animation Section
- 在‘Camera’标签中，选择‘GUI Camera’
- 在‘Move Mode’下锁定 Z 位置
- 在‘Object’标签中，选择‘Quest3D Logo’物体并将它移动到屏幕的右下角。
- 切换到运行模式（Run Mode）。
- 按住鼠标右键并拖动以绕着汽车旋转
- 单击复选框，注意到汽车停止旋转



③ 切换到编辑模式（Edit Mode）
③ 进入 Channel Section
③ 为‘R Slider Value’ channel 建立一个快捷方式并加它连接到‘Car’物体‘Car Chassis’面‘Emissive(发射)’channel 的‘R’子连接上。‘Car’ 物体位于‘3D Scene’文件夹中。
③ 进入 Animation Section
③ 切换到运行模式（Run Mode）。
③ 左右拖动‘R’滑块。注意汽车颜色的变化。
③ 切换到编辑模式（Edit Mode）。
③ 进入 Channel Section
③ 将‘Debug’ channel 连接到‘Project’ channel。注意一些统计数字显示在屏幕上。
③ 进入 Animation Section
③ 查看所有的统计数据：帧率（Framerate），视口的宽高，凹凸贴图（bump map）和顶点渲染（pixel shading）



完成后的场景

③ ..\Tutorials\2.7 – Graphic user interface\Graphic user interface – Complete.cgr

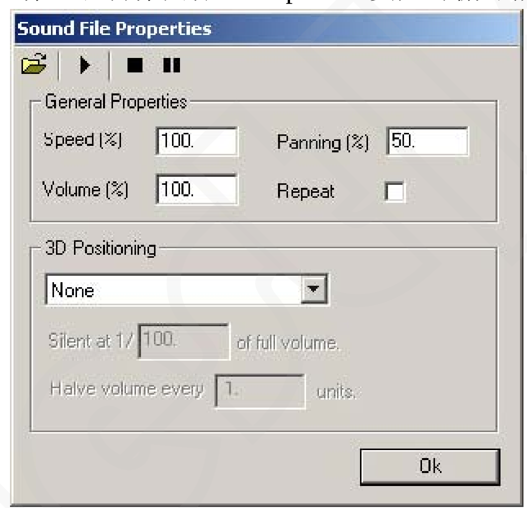
2.8 声音

从简单的单击菜单声音到完整的环境音乐，可以添加音频到场景中。就像绚丽的图形和栩栩如生的动画一样，声音有助于营造场景的气氛。带有声音的实例能够帮助用户更好的理解应用程序。如果在看电视或玩游戏的时候关掉声音你就会错过美妙的音频部分。

可以将声音以多种格式存放在计算机上，Quest3D 支持'.wav'和'.mp3'文件。

.Wav 格式

在 Quest3D 中可以在 Sound File channel 中存储'.wav'文件。在它的属性窗口中包含许多选项，例如加载，播放，停止和暂停图标。'Repeat'选贤允许循环播放。



速度范围从 0% 到 200%，音量范围从 0%到 100%。这两个选项都可以在它们对应的子 channel 中改变。Panning 控制左右声道 0%意味着只有左声道被使用，而 100%意味着只有右声道被使用。50%（缺省值）表示同时使用两个声道。

Sound Command 声音命令

Sound Command channel 允许控制声音的各个方面，例如回放和音量。也可以加载一个'.wav'文件。下拉列表给出了所有的选贤。一个声音命令必须被调用一次以便执行，并将影响 *Sound File* channel 的所有子 channel。

3D positioning（三维音效）

除了简单的播放声音以外，还可以将声音放置在三维空间中。产生三维音效需要一个 *Listener* channel。通常将它连接到像机的位置上，并且会根据像机和声源的相对位置自动调整音量，左右声道和频率。只有 Mono 类型的声音可以用于三维空间。*Sound File* channel 的'3D Positioning'选项定义模拟的精度，较高的精度需要更好的 CPU

'Silent at 1/* of full volume'是一个只考虑性能的分割点。'Half volume every * units'可以看作是声音的半衰值。这个值允许你模拟声音大小不同的声源。例如一个蜜蜂的声音半衰

值可能是 0.01 个单位，而飞机的音量可能是 10 个单位。

Listener 属性窗口包含三个高级选项。

- ③ **'Doppler Factor'**描述了声源在前后移动过程中的变化程度，缺省值 1 可以模拟真实世界的情况。
- ③ **'Rolloff Factor'**影响场景中所有声源的半衰值。缺省值 1 可以模拟真实世界中的情况，0 表示完全禁用，2 可以放大它的影响。
- ③ **'Distance Factor'**允许你调整不同的计量系统。如果 Quest3D 中的一个单位代表模型的一步，你可以将此值设置为 0.3048

在控制面板中检查'音频设备'以确保它被正确设置。

.MP3 格式

Qeust3D 也支持'.mp3'格式，*MP3 File* channel 用于存放这种类型的声音。通过它的属性窗口可以从硬盘上打开一个 mp3 文件。'Save File in Channelgroup'允许你将声音数据保存在该 channel 中。当然这会增加 channel 组文件的大小。*MP3File* 仅有的一个子连接可以以 *Text* channel 的形式包含文件名。

可以使用 *MP3 Control* channel 来控制回放和音量。还可以用于取得文件的长度，获取当前的播放位置，跳转到指定位置。

MP3 文件不能产生三维音效。

实例

Quest3D 场景

- ③ ..\Tutorials\2.8 – Sound and music\Sound and music.cgr

需要的声音

- ③ ..\Resources\Sounds\Bird.wav

需要的模版

- ③ Logic \ Channel Caller
- ③ Logic \ Trigger (x2)
- ③ Input \ UserInput (x2)
- ③ Sound \ Sound File
- ③ Sound \ Sound Command (x2)
- ③ Sound \ Listener

Step by step:

- ③ 打开'Sound and music.cgr'。该文件包含一个简单的场景和一个行走像机。
- ③ 添加一个 *Sound Command* channel 并双击打开属性对话框。从下拉列表中选择'Play'

命令并单击‘OK’按钮。将 ‘Sound Command’ channel 连接到‘Play Sound’ channel。

- ③ 添加一个 *Sound File* channel 到 Channel 视图中并将它连接到 Sound Command。双击打开它的属性对话框。单击 Load 图标，找到‘Car engine.wav’文件。单击‘Open’打开它。
- ③ 确定 Animation 3D View 窗口是打开的。
- ③ 切换到运行模式（Run Mode）。
- ③ 单击空格测试场景。注意 Sound Command 被触发，声音开始播放。
- ③ 切换到编辑模式（Edit Mode）。
- ③ 添加另一个 *Sound Command* channel 并将它连接到‘Stop Sound’ channel。设置该 Sound Command 为‘Stop’。连接此 Sound Command 到‘Stop Sound’ channel。
- ③ 将原来的‘Sound File’ channel 连接第二个 *Sound Command* channel
- ③ 切换到运行模式（Run Mode）
- ③ 单击空格以测试场景，一段时间后，单击‘Backspace’键，注意声音停止播放了。
- ③ 切换到编辑模式（Edit Mode）
- ③ 双击 *Sound File* channel。从 3D Positioning 下拉列表中选择‘Normal’。钩选‘Repeat’以便重复播放。
- ③ 创建‘Car’ channel 中‘Motion’ channel 的快捷方式，并将它连接到 *Sound File* channel 的第一个子连接上。现在声源连接到了一个三维模型上。
- ③ 拖动一个 *Listener* 到 Channel 视图中，并将它连接到‘Audio’ channel
- ③ 创建‘Walkthrough Camera’ channel 中 Motion 的快捷方式，将它连接到 *Listener*。
- ③ 确定 Animation 3D View 窗口显示当前场景并使用了投影像机（Project Camera）。
- ③ 切换到运行模式（Run Mode）。
- ③ 单击‘P’播放声音。
- ③ 使用方向键移动三维物体，注意声音的变化。
- ③ 使用鼠标旋转视角，注意左右声道声音的变化。



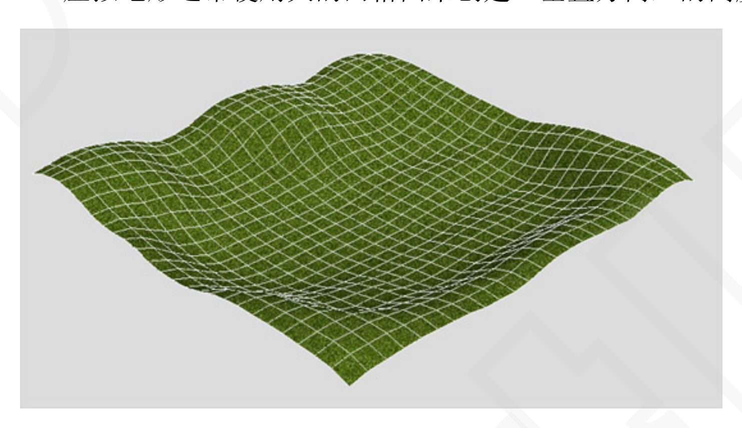
完成后的场景

- ③ ..\Tutorials\2.8 – Sound and music\Sound and music – Complete.cgr

2.9 地形和环境

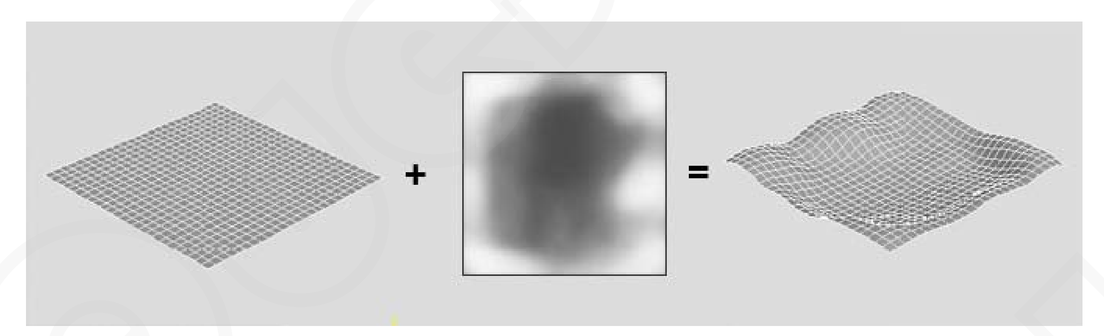
虚拟外部环境是非常有用的。它们可能相当复杂而导致较低的帧率。本章讨论了如何创建一个环境并提供一些有效的优化方法。

虚拟地形通常使用大的网格面来创建。垂直方向上的高度(Y 值)定义了高山和峡谷。



Height maps（高度图）

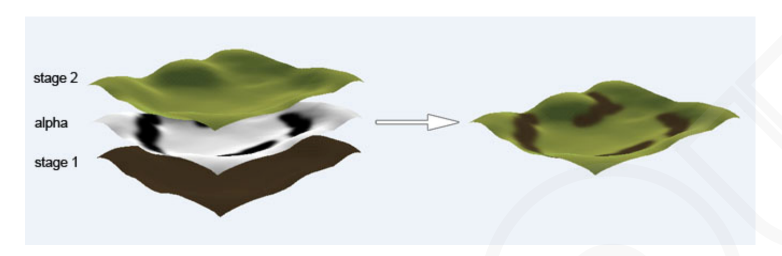
大多数三维建模软件如 3D Studio Max 和 Maya 都允许你直接或间接的创建地形。一个灰度图可以用来定义组成地面的垂直方向的 Y 值：黑色表示较低的点，白色为较高的点。



可以在你使用的建模软件的帮助手册中搜索关键字‘height map’，‘displace’以获取更多信息。

贴图

可以将一个或多个贴图应用到一个面上。不同的‘texture stages’可以借助 alpha 贴图来相互混合。



Multiple passes（多重通道）

2.4 节‘Surface properties’讨论了 Quest3D 不同的混合方法。要达成上述目的并使其能够运行在较老的显卡上，建议使用多重通道。这种方法可以多次渲染同一个面，并使用透明度来混合这些实例。

渲染一个典型的地形表面可能包含如下步骤：

- ③ 第一层，如泥土（UV set 0）
- ③ 第二层，如草地(UV set 0)，通过一个 alpha 贴图来混合(UV set 1)
- ③ 光照贴图层，使用‘multiply’操作(UV set 1)

天空球

虚拟世界户外场景的天空通常使用半球来表示。



光照

可以使用一个强烈的方向光模拟太阳光。此外可以增加场景中所有三维模型的 Emissive 值，显得更加明亮，光照贴图和实时阴影能够极大地改进室外场景。关于光照和阴影参见 2.5

雾

雾可以营造一种气氛以显示景深。使用它的好处可以减小像机的远裁减面从而提高帧率。参见 2.6

自然景观

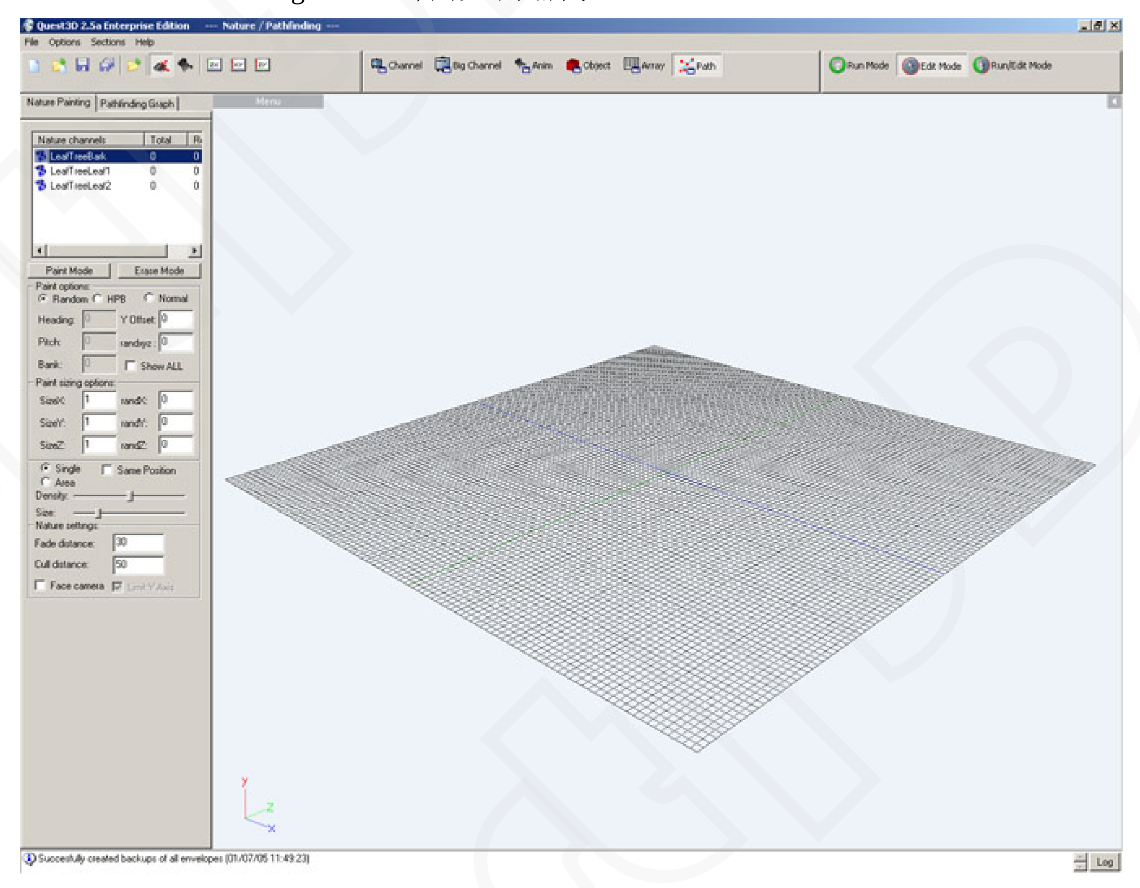
大多数室外场景中有大量的树和植物。Quest3D 有一个名为‘Nature Painting’的高级特性。它允许你非常容易的添加大量的树和植物到场景中。Nature Painting 系统针对性能进行了优化并使用硬件来快速高效的渲染场景。



Nature Generator channel 被放置在物体的 *Surface* channel 和 *3D ObjectData* channel 之间。它包含了所有实例的位置，旋转和缩放还有一些扩展信息。需要与场景中的植物发生碰撞的物体必须连接到 *Nature Generator*

‘Nature / Pathfinding’ Section 提供了使用这些特性的简单方法。

Nature / Pathfinding Section 布局如下图所示：



Nature Painting 属性

通过‘Nature Painting’标签中的选项，能够定义物体的所有方面，外观如旋转和大小这些值也可以是随机的。可以放置单个物体或者使用‘Area’选项将它们作为一个组放置。‘Cull distance’值意味着在这个距离之外的物体将会被从像机的视野中剔除。‘Fade distance’值可以用于将实例与环境混合在一起，以掩饰一些荒芜的地方。要实现这种效果，首先必须正确设置透明方式。正确的设置是‘Alpha Value 和 Texture’，可以在 Object Section 的‘Surface’标签中找到这些选项

‘Face camera’选项能够产生非常好的效果，但是会使用较多 CPU 资源。建议只将它用于简单的 Nature Paint 物体上，例如一个显示草的平面上。

模版

Quest3D 预定义了大量的地形，树和植物。可以从模版中直接使用。

实例

Quest3D 场景：

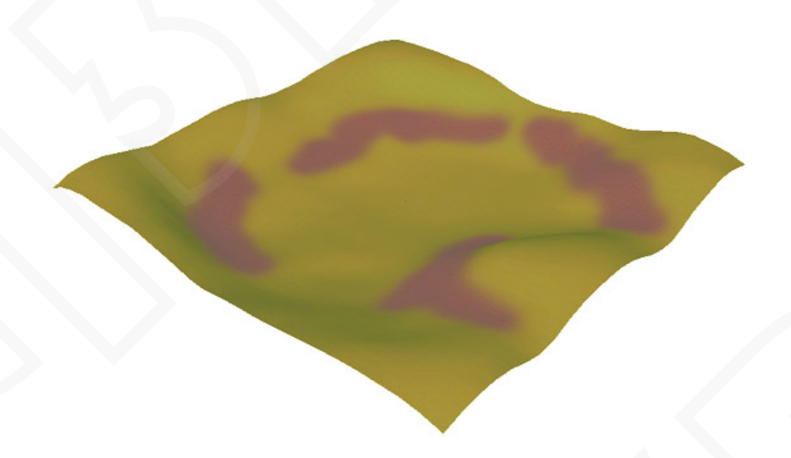
- ..\Tutorials\2.9 – Landscapes\Landscapes.cgr

需要的模版

- Objects \ Object Data \ Nature Generator

Step by step:

- 打开文件‘Landscapes.cgr’，该文件包含一个地形，一个用于 nature painting 的草和树的模型。同时还包含一个天空球和一个行走像机。
- 进入 Object Section
- 在列表中单击‘Landscape’物体
- 在列表中单击‘Dirt Layer’面选中它。
- 进入右侧的‘Texture Stages’标签。
- 确定‘Stage 1’被选中。
- 进入右侧的‘Textures’标签，注意当前使用的贴图高亮显示。双击这个贴图以打开它的属性对话框。
- 单击‘Load’按钮进入‘..\Resources\Textures\Landscapes\’文件夹。在此文件夹中包含许多地形的贴图。选择一个泥土的类型，然后单击‘Open’。注意新的贴图显示在预览窗口宗。单击‘OK’关闭贴图窗口。
- 在列表中单击’Grass Layer’以选择它。
- 在‘Texture Stages’列表中选择‘Stage 1’。在‘Textures’列表中双击高亮显示的贴图打开它的属性对话框。
- 单击‘Load’按钮从该目录下选择一个草的贴图。单击‘Open’。 注意新的贴图显示在预览窗口宗。单击‘OK’关闭贴图窗口。
- 在‘Texture Stages’标签中选择‘Stage 2’ 在‘Textures’列表中双击高亮显示的贴图打开它的属性对话框。
- 单击‘Load Alpha’按钮从该目录下选择一个‘Alpha’贴图。单击‘Open’。 注意新的贴图显示在预览窗口宗。单击‘OK’关闭贴图窗口。
- 进入左边的‘Surface’标签
- 选择‘Alpha Texture’的透明度。注意地形的两个贴图层现在相互混合了。



- ③ 进入 Nature / Pathfinding Section
- ③ 单击‘Grass Nature Generator’物体选中它
- ③ 单击‘Paint Mode’按钮
- ③ 在 Animation 3D View 窗口中的地形上移动鼠标。注意物体随着你的鼠标移动并会自动放置在正确的高度上。单击鼠标左键放置一个草的实例，多放置一些草在场景中。
- ③ 选择‘Face camera’选项，和‘Limit Y axis’。注意场景中的实例是如何朝向像机的。
- ③ 设置‘Fade distance’为 0 设置‘Cull distance’为 50。注意远处的实例，远处的实例与场景混合在一起，超过 50 个单位的实例不会显示。
- ③ 再次单击‘Paint Mode’退出
- ③ 进入 Channels Section
- ③ 选择并删除‘3D Object Data’ channels 和‘Tree Leaves’ channels 之间的连接。
- ③ 添加一个 *Nature Generator* channel 并将它连接到‘Tree Leaves’ channel。重命名为‘Tree Leaves Nature Generator’
- ③ 将‘3D Object Data’ channel 连接到‘Tree Leaves Nature Generator’ channel
- ③ 创建‘Collision Object’ channel 的快捷方式并将它连接到‘Grass Nature Generator’ channel 和‘Tree Leaves Nature Generator’ channel 的‘Collsion Object’子连接上。现在‘Tree’物体以可以用于绘制了
- ③ 进入 Nature / Pathfinding 标签
- ③ 单击‘Tree Leaves Nature Generator’物体选择它。
- ③ 按住’Ctrl’键单击‘Tree Trunk Nature Generator’物体加选它
- ③ 单击‘Paint Mode’按钮
- ③ 在‘Paint sizing options’菜单中，设置‘randX’, ‘randY’和‘randZ’得值为 0.5。放置一些树，注意它们大小的变化。
- ③ 选择‘Area’项，设置‘Size’为 250，‘Density’为 15%，单击一下放置更多的树。
- ③ 再次单击‘Paint Mode’退出
- ③ 进入 Animation Section.
- ③ 确定 Animation 3D View 窗口中是使用投影像机（Project Camera）显示当前场景。
- ③ 切换到运行模式（Run Mode）
- ③ 使用方向键和鼠标在场景中行走以测试场景。



完成后的场景：

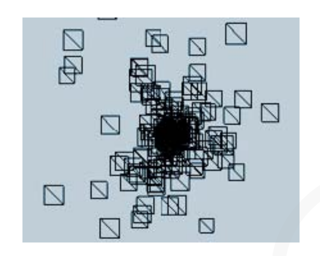
③ ..\Tutorials\2.9 – Landscapes\Landscapes – Complete.cgr

2.10 粒子系统

许多电影和游戏中的特效，如火，烟，爆炸和所有类型的魔法效果都是由粒子系统创建的



粒子系统是从一个发射器释放的物体的集合并且会随时间而变化。粒子是面向像机的平面物体



Motion（运动）

对于该系统中的每个粒子，它们的位置，旋转和大小都被保存。此外，其他的属性例如，速度，颜色和透明度也会被使用。作用力例如重力也会影响系统中的所有粒子。

Emitter（发射器）

粒子是从发射器中发射的。Quest3D 中，发射器是 *3D Object Data* channel 的一种形式。粒子系统循环物体上所有的点并不停的从其中一个位置上释放粒子。通过随机的在三维空间中放置一些点，就像是粒子在随机释放一样。这样就可以产生非常好的特效例如火和烟。

面属性

粒子的外观可以通过设置平面物体 Diffuse 和 alpha 贴图来定义。参看 2.4 中关于面属性和混合方法的介绍。

类型

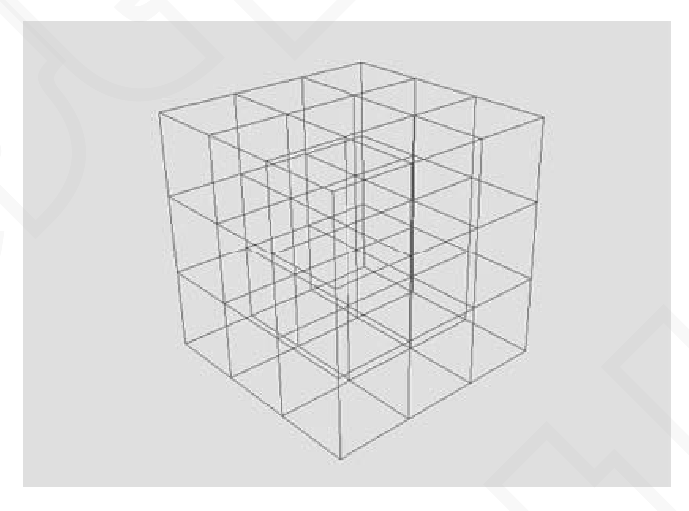
Quest3D 中有两种类型的粒子系统。*Particle Object* channel 是一个小型的系统，并可以用于简单的效果例如火。

高级的粒子系统包含大量的 channel 并能被用于模拟停留在碰撞物体内部的烟

高级的粒子系统允许更多控制属性例如速度，旋转，大小和颜色。表示函数的 *Envelope* 可以被连接到 *Particle Emitter* channel。可以用'Advanced Function Base'得值作为这些 *Envelope* 的一个索引。

Particle Flow Grid（粒子流向网格）

可以用 *Particle Flow Grid* channel 来控制粒子系统的流向。将一个碰撞物体划分成不同的部分，在每一个部分内计算力。作用力可以由碰撞物体，外部物体或者是粒子自身产生。



粒子的碰撞是作为点碰撞产生。因此，粒子的面可以依附于碰撞物体上。使用较小的物体作为碰撞物体是一种良好的做法。并且，简化的碰撞物体也可以提高性能。

Reference Manual

Quest3D 参考手册中详细描述了各种高级粒子系统 channel。

实例

Quest3D 场景:

③ ..\Tutorials\2.10 – Particle systems\Particle systems.cgr

需要的模版:

③ Objects \ Particles \ Simple \ Small Fire
③ Objects \ Particles \ Advanced \ Smoke
③ Objects \ Particles \ Helpers \ Particle Flow Grid
③ Objects \ Collision \ Collision Object
③ Variables \ Value \ Value
③ Variables \ Vector \ Vector with Envelopes

Step by step:

③ 打开文件‘Particle systems.cgr’。该场景中包含一个半球形的环境物体和一个不可见的小碰撞物体。
③ 拖动一个 *Small Fire* 模版到 Channel 视图中并将它连接到‘Render Scene’channel。注意火的效果马上就可以在 Animation 3D View 窗口中看见
③ 添加一个 *Smoke* 模版并将它连接到‘Render Scene’channel。注意烟的效果马上就可以在 Animation 3D View 窗口中显示出了。同时注意烟粒子没有限制的穿过了环境物体的表面。
③ 改变 position 向量中的 Y 值为 1
③ 拖动 *Particle Flow Grid* channel 到 Channel 视图中并将它连接到‘Smoke’物体的‘Particle Emitter’ channel 上
③ 添加一个 *Collision Object* channel 并将它连接到‘Particle Flow Grid’ channel
③ 连接‘Interior Collision Object’ channel 到 *Collision Object* channel
③ 双击‘Particle Flow Grid’ channel 打开属性对话框。设置‘Resolution’的‘x’, ‘y’和‘z’的值为 10。选择‘Show Grid’选项并单击‘Generate Grid’按钮。注意一个网格显示在 Animation 3D View 窗口中（如果‘Dynamic objects’选项被打开）。单击‘OK’
③ 双击‘Particle Emitter’ channel 打开它的属性窗口。在‘Grid’标签中，选择‘Particles Are Affected By Grid’选项。注意在 Animation 3D View 窗口中烟粒子受到了网格的影响。单击‘OK’



③ 拖动一个 *Value* channel 到 Channel 视图中并将它连接到‘Particle Emitter’ channel 的‘Advanced Function Base’字连接上。重命名该 Value 为‘Advanced Function Base’
③ 添加一个 *Vector (using envelopes)*模版并将它连接到‘Particle Emitter’ channel 的‘Particle Color Function’字连接上。分别重命名其中的 *Envelope* 为‘Red’, ‘Green’ 和‘Blue’
③ 创建‘Advanced Function Base’ channel 的快捷方式并将它连接到每一个 *Envelope* 上
③ 双击‘Red’ *Envelope* 打开它的属性对话框。在(0, 0.5)添加一个键并单击‘OK’
③ 双击‘Green’ *Envelope* 打开它的属性对话框。在(0, 0.5)添加一个键并单击‘OK’
③ 双击‘Green’ *Envelope* 打开它的属性对话框。在(0, 0.5)和(50, 0)处分别添加一个键，并单击‘OK’ 注意粒子将在他们的生存期内变蓝。

完成后的场景：

③ ..\Tutorials\2.10 – Particle systems\Particle systems – Complete.cgr

2.11 角色动画

2.2 介绍了三维模型的动画。在本节中，将讨论动画的高级形式：角色动画。

人和动物在移动过程中不仅仅是简单的滑动。他们身体的各个部分都是不相关的。但是，又不是一些独立分支的集合，这些部分都是相对于其他部分而移动的。但一个部位移动时，皮肤会沿着它拉伸。

虚拟骨骼

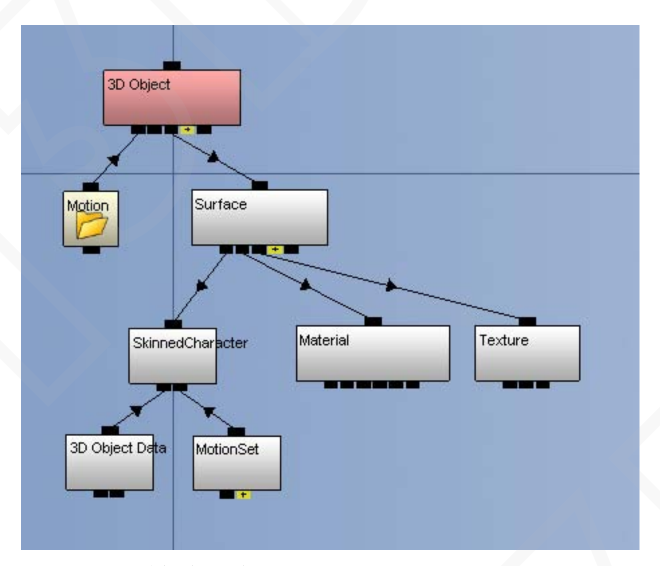
可以使用相似的方式为三维模型设置动作。模型的多边形面可以看作皮肤。可以用虚拟骨骼使皮肤各个部分动起来。实际说话过程中，三维模型上的每一个点必须连接到适当的虚拟骨骼上。



高级的建模软件例如 3D Studio Max 带有预定义的虚拟骨骼系统。可以直接将它连接到你的角色上，还可以调整和扩展它。

Quest3D 中的角色动画

虚拟骨骼动画数据可以被导入 Quest3D 中。一个带有骨骼的三维模型动画系统与 *3D Object* channel 具有类似的结构。还添加了两个 channel。



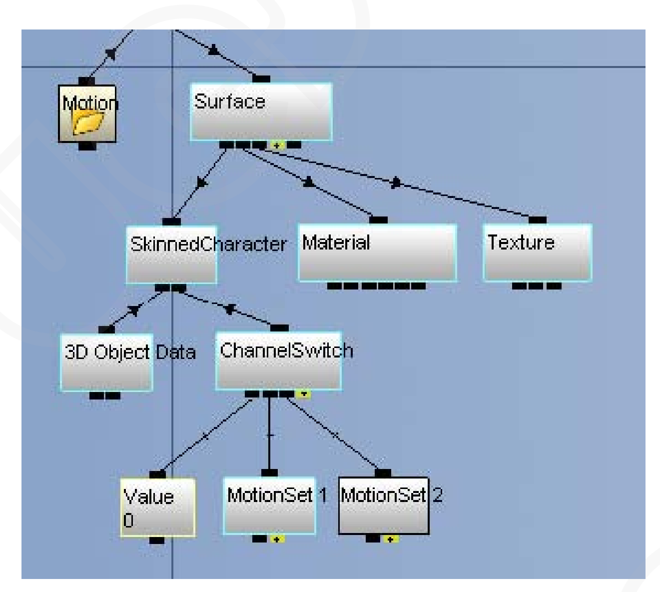
Skinned Character channel 被放置在 *Surface* channel 和 *3D Object Data* channel 之间。它包含了虚拟骨骼的定义和三维模型如何被连接到虚拟骨骼上的相关信息。

Motionset channel 以关键帧的形式包含了所有的骨骼动画数据，其中包含位置，旋转和缩放。

可以通过连接一个 *Value* 到 *Motionset* 的第一个子连接上来播放动画（例如，一个 *TimerValue*）

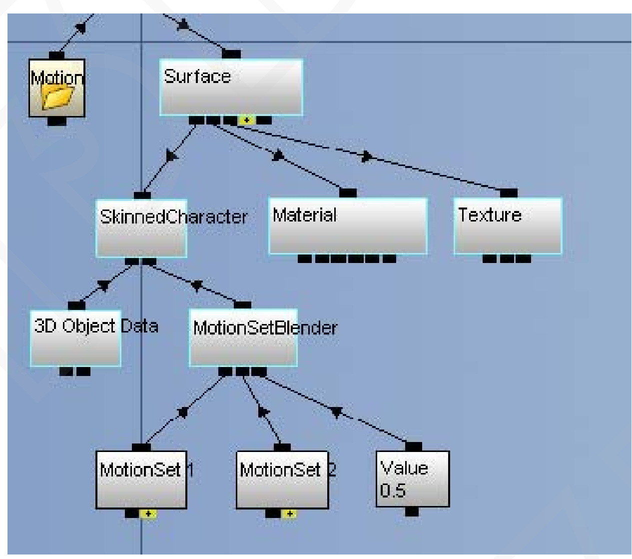
切换 Motionset

场景中的角色可能有多个动作。在任意给定的时间点上，它可以站立，走，跑或者使用物体。在这些动作间切换的方法是使用一个 *Channel Switch*，可以基于索引值来选择 *Motionset*。



Motionset Blender

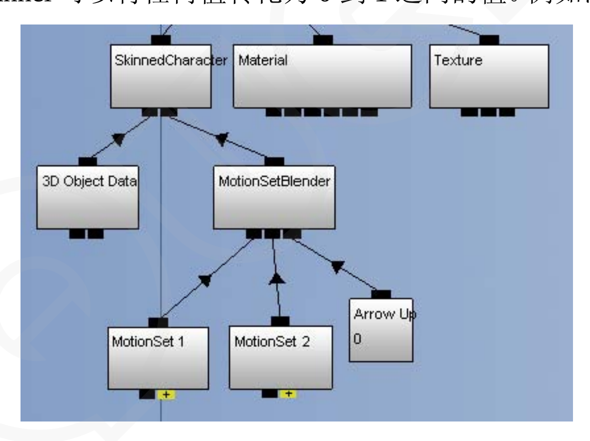
上述方法会导致不连贯的过渡。为使动作之间更加平滑的过渡，可以使用 *Motionset Blender*，这个 channel 接受两个 *Motionset* 和一个 *Value*。这个 *Vlaue* 定义了如何显示这两个动作。0 意味着只有第一个 *Motionset* 被使用，1 意味着只有第二个被使用。介于 0 和 1 之间的值表示使用线性插值的方式来混合两个动作。

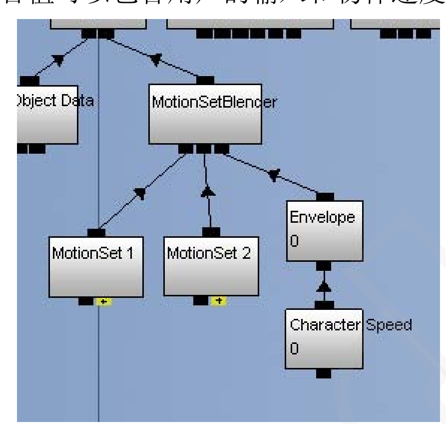


下图分别显示了 100% 走, 50% 走和坐, 100% 坐



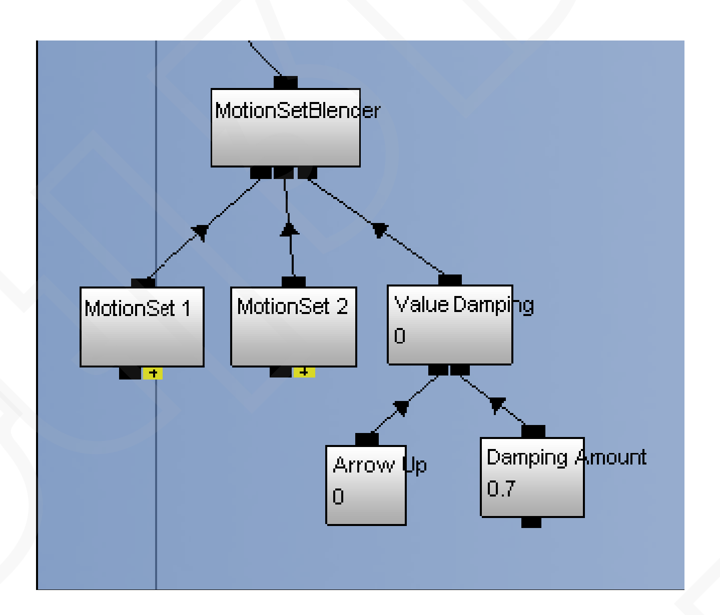
Motionset Blender 的混合值可以来自于 Quest3D 程序中的任何类型的信息。*Envelope* channel 可以将任何值转化为 0 到 1 之间的值。例如混合值可以包含用户的输入和物体速度。





Value Damping(值衰减)

动态从 0 过渡到 1 或者从 1 过渡到 0 的一种简洁而有效的方法是使用 *Value Damping* channel



实例

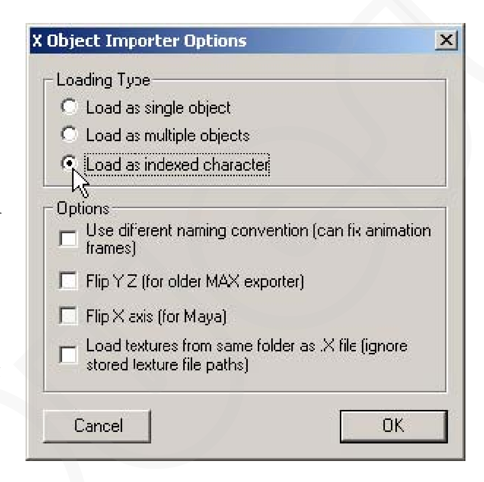
下面的练习解释了如何将角色动画导入到 Quest3D 中。同时描述了动作混合。

需要的 .X 文件:

③ ..\Resources\Characters\Animated character – Idle.X
③ ..\Resources\Characters\Animated character – Walk.X

需要的模版:

③ Scene \ Start 3D Scene
③ Scene \ Render Camera Light
③ Timing \ TimerValue
③ 3D Items \ Object Data \ Motionset Blender
③ Variables \ Value \ Value Damping
③ Logic \ User Input \ User Input



Step by step:

③ 从文件菜单中选择‘Import’
③ 找到文件‘..\Resources\Characters\Animated character – Idle.X’单击‘Open’
③ 使用‘Character animation 1’作为 Pool 的名称，单击‘OK’
③ 在‘X Object Importer Options’窗口中，选择‘Load as indexed character’，单击‘OK’。现在

这个三维角色模型就被导入到了 Quest3D 中了。

③ 重命名 3D Object channel 为‘Animated Character’

③ 拖动一个 *Start 3D Scene* channel 到 Channel 视图中。右键单击并选择‘Set as start channel’。

③ 添加一个 *Render Camera Light* 模版并将‘Render’ channel 连接到‘Start 3D Scene’ channel。

③ 将‘Animated Character’ channel 连接到‘Render’。注意角色将显示在 Animation 3D View 窗口中。



③ 重命名‘Motionset’ channel 为‘Idle Motionset’

③ 拖动一个 *Timer Value* 到 Channel 视图中并将它连接到‘Idle Motionset’ channel。双击‘Timer Value’ channel 从下拉列表中选择‘New..’在编辑框中输入‘Animated Character Idle’单击‘OK’关闭属性对话框。

③ 进入 Animation Section

③ 使用投影像机（Project Camera）在 Animation 3D View 窗口中查看该场景。

③ 从 timer 下拉列表中选择‘Animated Character Idle’。设置结束帧为 450 并单击 Play 按钮。注意角色开始移动

③ 进入 Channel Section

③ 删除‘Idle Motionset’ channel 和 ‘Skinned Character’ channel 之间的连接

③ 添加一个 Motionset Blender 并将它连接到‘Skinned Character’ channel

③ 连接‘Idle Motionset’到 Motionset Blender

③ 使用与上面相同的步骤导入‘Walk.X’

③ 找到新导入的‘Motionset’ channel 并复制它

③ 切换到主 channel 组

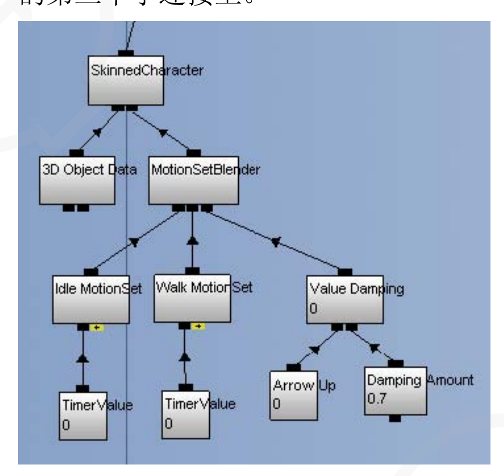
③ 将 *Motionset* 粘贴到 Channel 视图

③ 重命名该 *Motionset* 为‘Walk Motionset’并将它连接到‘Motionset Blender’ channel 的第二个子连接上

③ 拖动另一个 *Timer Value* channel 到 Channel 视图中并将它连接到‘Walk Motionset’ channel

③ 重命名 *Timer Value* 为‘Animated Character Walk’

③ 拖动一个 *Value Damping* 到 Channel 视图中并将它连接到‘Motionset Blender’ channel 的第三个子连接上。



- ③ 添加一个 *User Input channel* 并将它连接到‘Value Damping’ channel。设置 *User Input* channel 为向上的方向键
- ③ 进入 Animation Section
- ③ 从 Timer 下拉列表中选择‘Animated Character Walk’设置结束帧为 27 并单击 Play 按钮
- ③ 切换到运行模式（Run Mode）
- ③ 按向上方向键测试场景。注意 *Value Damping* 随着时间从 0 到 1 增加。同时注意三维角色从空闲动作到走动作的过渡。
- ③ 释放向上方向键。注意 *Value Damping* 随着时间从 1 到 0 减小。同时注意三维角色从走动作到空闲动作的过渡。
- ③ 切换到编辑模式（Edit Mode）。

完成后的场景：

- ③ ..\Tutorials\2.11 – Character animation\Character animation 1 – Complete.cgr

可以保存当前 channel 组，下面的步骤将开始一个新的场景。

Quest3D 场景:

- ③ ..\Tutorials\2.11 – Character animation\Character animation 2.cgr

需要的.X 文件:

- ③ ..\Resources\Characters\Animated character – Idle.X
- ③ ..\Resources\Characters\Animated character – Walk.X
- ③ ..\Resources\Characters\Animated character – Run.X

需要的模版:

- ③ Variables \ Value \ Value Damping
- ③ Variables \ Value \ Envelope (x2)

Step by step:

- ③ 打开文件‘Character animation 2.cgr’。它包含一个简单的场景，一个具有多个动作的角色和一个第三人称像机。
- ③ 进入 Animatin Section
- ③ 在 Animation 3D View 窗口中使用投影像机（Project Camera）查看该场景。
- ③ 切换到运行模式（Run Mode）
- ③ 使用方向键和鼠标在场景中移动角色。注意像机总是位于角色的后面。现在角色还不能正确地移动并且只显示了空闲动作序列
- ③ 切换到编辑模式（Edit Mode）
- ③ 进入 Channels Section
- ③ 拖动一个 *Value Damping* 模版到 Channel 视图中并将它连接到‘Motionset Blender’ channel 上，该‘Motionset Blender’ channel 上已经连接了‘Idle’ channel 和‘Walk’ channel
- ③ 添加一个 *Envelope* channel 并将它连接到‘Value Damping’ channel
- ③ 创建一个‘Character Speed’ channel（碰撞响应系统的一部分）快捷方式并将它连接到 *Envelope*

- ③ 双击 *Envelope*。在(0, 0)位置添加一个点。该点意味着角色的速度（输入）为 0 时，*Motionset Blender* 将完全使用‘Idle’动作。
- ③ 在(0.02,1)处添加一个点，意味着角色的速度为 0.02 时，*Motionset Blender* 将完全使用‘Walk’动作。如果速度在 0 到 0.02 之间的时将导致‘Idle’ 和‘Walk’动作的混合。
- ③ 切换到运行模式（Run Mode）
- ③ 按向上的方向键测试场景。注意‘Character Speed’ channel 值的改变，角色‘Idle’ 到‘Walk’动作的混合。
- ③ 释放向上的方向键，注意‘Character Speed’ channel 值的改变，角色‘Walk’到 ‘Idle’动作的混合
- ③ 切换到编辑模式（Edit Mode）
- ③ 添加另一个 *Envelope* 并将它连接到 *Value Damping* channel，此 channel 已经连接了‘Walk’ channel 和 ‘Run’ channel
- ③ 创建一个‘Character Speed’ channel 的快捷方式并将它连接到这个新的 *Envelope*
- ③ 双击该 *Envelope*。在位置(0.25, 0)和(0.5, 1)位置处添加两个点。
- ③ 进入 Animation Section.
- ③ 切换到运行模式（Run Mode）
- ③ 按上方向键测试场景。注意角色开始走动。
- ③ 同时按 shift 和上方向键，注意角色的速度增加了并开始跑。



完成后的场景：

- ③ ..\Tutorials\2.11 – Character animation\Character animation 2 – Complete.cgr

第三部分：编程

摘要

3.1：逻辑

本章描述逻辑的概念——项目的流程。

3.2：数学

在 Quest3D 项目中数值是经常变化的。数学可用于计算新值。

3.3：循环

每帧中一个 For Loop 执行程序特定部分多次。

3.4：数组

数组可以存储多个特定类型的数据。

3.5：多 channel 组

高级 Quest3D 程序可能是复杂的项目。 在某些情况下可以将一个程序划分为多个部分

3.6：数学运算符

所谓的 Operator channel 允许你动态地改变每一个变量。

3.7：寻径

角色可能不得不停止在交通灯，断开的桥前面或者彼此响应。寻径允许以某种动态的线路选定。

3.8：有限状态机

有限状态机 channel 可用于简化复杂结构中子程序之间切换。

3.1 逻辑

在 Quest3D 中,逻辑是程序流程。逻辑描述发生什么和什么时候发生。

逻辑的例子：

③ 一个物体是不是显示在屏幕上。

③ 按一个键打开一个门。

③ 时间事件触发音乐。

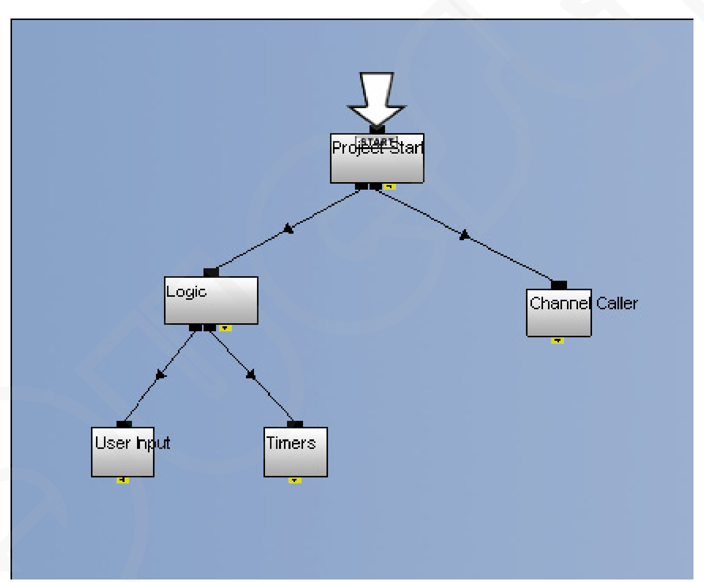
③ 根据一个菜单选择当前的应用程序屏幕。

许多 Quest3D channel 特别地为将逻辑添加到程序而设计。 这些逻辑 channel 中最主要的将在本章论述。

Channel Caller

如同他们的名称，Channel Caller 简单地调用与他们有关系的 channel。 通常用它们将结构添加到工程。 在 Quest3D 程序中可以把 Channel Callers 看做是分配器，甚至 ‘chapter’。

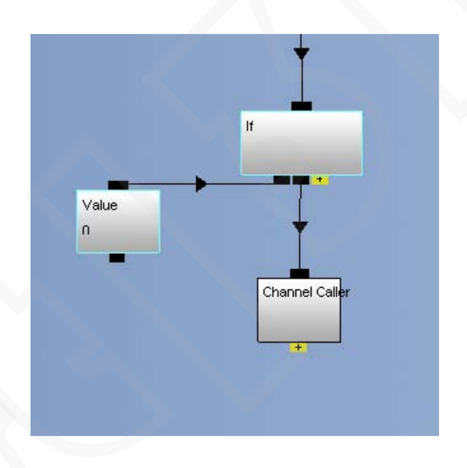
下图显示一个由 Channel Callers 划分的程序结构的例子。

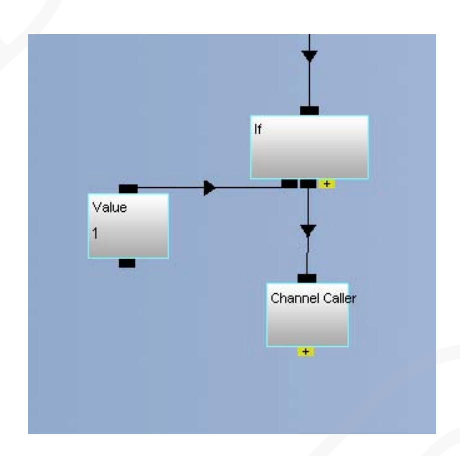


应该适当地命名和经常使用 Channel Callers。

If

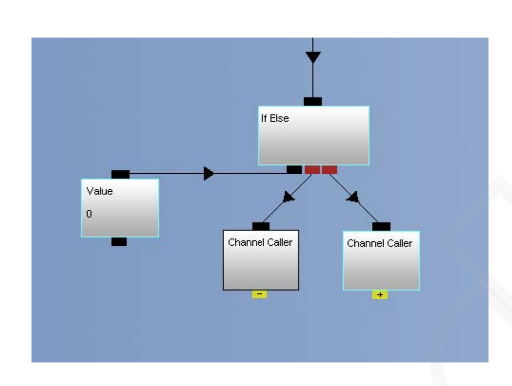
如同 Channel Caller 一样，If channel 也是调用它的子 channel。然而只有当满足某种情况时 If channel 才会调用它的孩子。需要判断的条件可以连接到 Ifchannel 第一个子连接块的，并且必须是 value。只有当该值不为’0’时（条件为’true’），连接到 Ifchannel 的 channel 才会被调用。

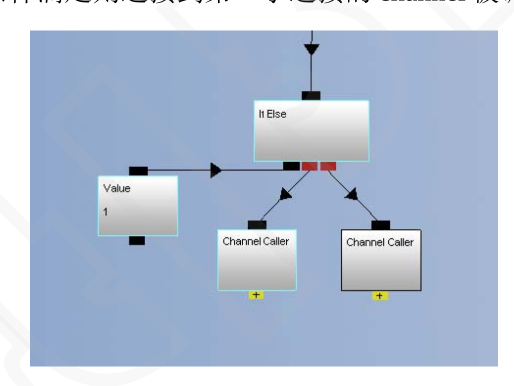




If Else

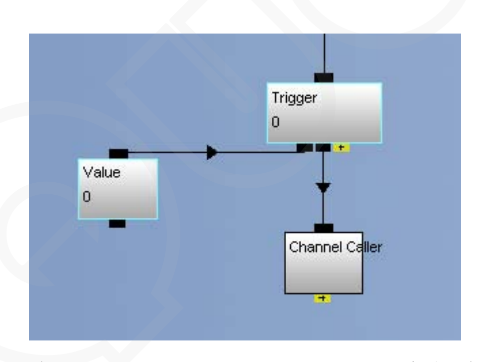
Ifelse channel 类似于 Ifchannel。如果连接的条件不满足（换言之,值等于‘0’）,那么连接到第三个子连接的 channel 被调用。如果条件满足则连接到第二子连接的 channel 被调用。

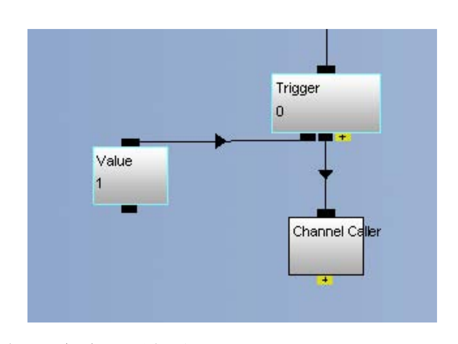




Triggers（触发器）

Trigger channel 的作用类似于 ifchannel。Trigger 也是根据一个条件调用他们的子 channel。

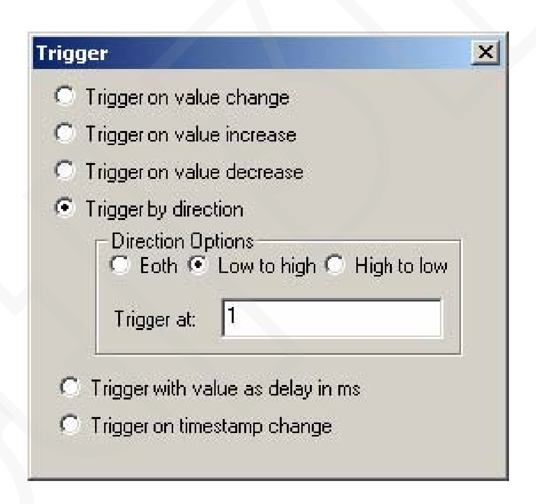




在 If channel 和 Trigger channel 之间存在两个主要差异。

第一，当条件满足的时候 Trigger 仅执行他们的子 channel 一次。对于一个需要再次激活的 Trigger,条件将不得不首先变为 false,然后变为 true。例如：一个 Trigger 被设置为按空格键触发，为了使它再次触发，空格键必须首先释放，然后按再一次。

第二个差异涉及到输入条件。除了用‘0’或‘1’的形式,还可以是属性窗口中列出的选项中的一个：



User Input（用户输入）

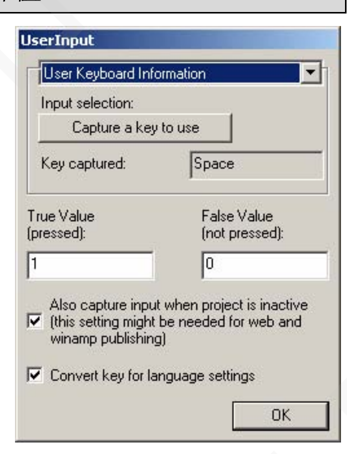
User Input channel 能够记录许多来源的动作例如键盘和鼠标。在 Quest3D 中，用户输入产生三种类型之一：

一个二元值‘0’或‘1’。例如：一个键或鼠标按纽按下。

一个从‘0’到‘1.0’的模拟值。例如：游戏杆的移动。

一个屏幕坐标值。例如：鼠标移动。

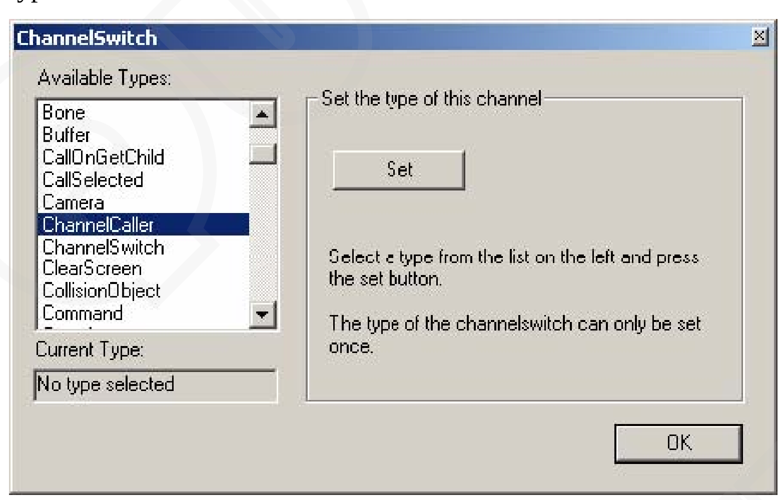
用户输入可以充当 If (Else) channel 和 Trigger channel 的条件值。



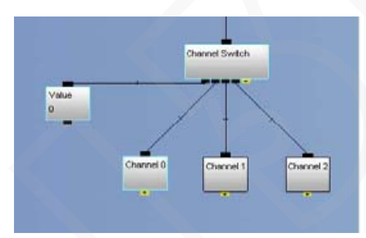
在属性窗口中，键盘上的任何键都可以绑定到一个 User Input channel。单击‘Capture a key to use’按钮后,按下你选择的键。类似,任何鼠标或游戏杆的按钮和移动都可以被记录。

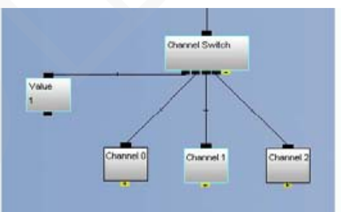
Channel Switche

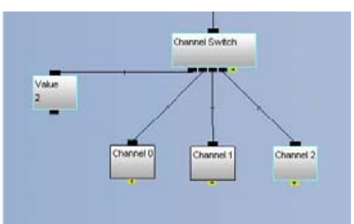
Channel Switches 允许你在任何类型的不同 channel 之间选择。Channel Switch 必须首先设置与它的子 channel 相同的 base type。



其次的,它的第一个子连接定义使用哪个 channel：值‘0’代表第一个 channel 被使用,‘1’为第二个,等等。







实例

在 Quest3D 中逻辑定义你的项目流程。本实例练习使用 Channel Caller 建立一个程序,并利用条件。条件作为 Ifand Trigger channel 的输入。

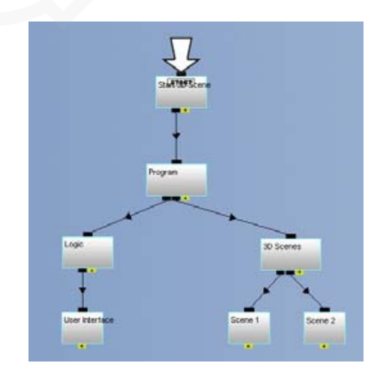
在完成这个实例以后，你将对 Quest3D 中的逻辑有一个更透彻的理解。

需要的模板：

③ 3D Items \ Start 3D Scene
③ Logic Channel Caller (x6)
③ Logic \ if
③ Logic \ Trigger
③ Input \ Input: Arrow left
③ Input \ Input: Arrow right
③ Scene \ Render, Camera, Light
3D Items \ Box

Step by step:

③ 拖动 Start 3D Scene 到 Channel 视图上。右键单击并选择‘Set as start channel’。
③ 添加一个 Channel Caller 并将它连接到‘Start 3D Scene’channel。
③ 重命名这个 Channel Caller 为‘Program’。
③ 拖动五个或更多 Channel Caller 到 Channel 视图之上。如下重命名他们：‘Logic’,‘User Interface’,‘3D Scenes’,‘Scene 1’和‘Scene 2’。
③ 为你的程序考虑一个逻辑结构并连接这个五个 Channel Caller。
③ 当完成时，查看下图的 channel 结构是否你自己答案看上去与它一样。



③ 现在，‘Scene 1’ 和 ‘Scene 2’ 两个都被调用。下面的步骤将添加一个逻辑并以键盘输入为根据决定使用显示那一个。

③ 断开 ‘Scene 1’ 和 ‘3D Scenes’ channel 之间的连接。

③ 添加一个 Ifchannel 并将它作为 ‘3D Scenes’ channel 的第二个子连接。

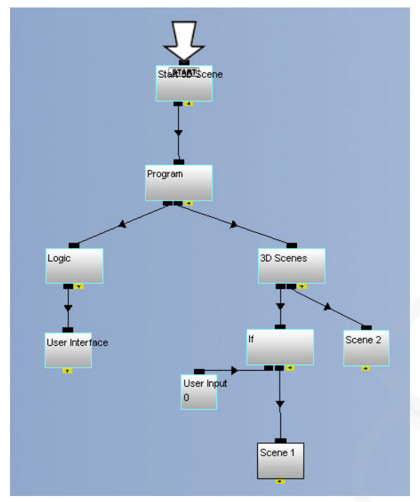
③ 连接 ‘Scene 1’ channel 到 ‘if’ channel 的第二个子连接。注意 ‘Scene 1’ channel 没有被调用（没有可见的蓝线）。

③ 拖动一个 Value 到 Channel 视图上并把它连接到 ‘if’ channel 的第一个子连接上。该 Value 将成为 if 的条件。

③ 改变这个 Value channel 的为值 ‘1’。注意 ‘Scene 1’ channel 被调用。

③ 改变 Value channel 的值为 ‘0’。注意 ‘Scene 1’ channel 不再被调用。

③ 选择 ‘Value’ channel 并删除它。



③ 添加一个 User Input channel 并把它连接到 ‘if’ channel 的第一个子连接上。当前的 channel 结构看上去与下图类似。

③ 双击 ‘User Input’ channel 打开它的属性。点击 ‘Capture a key to use button’ 并按 ‘1’ 键绑定它。单击 ‘OK’ 按钮。

③ 按按钮切换到 Run Mode。

③ 按住 ‘1’ 键注意 ‘Scene 1’ channel 被调用。

③ 释放 ‘1’ 键注意 ‘Scene 1’ channel 不再被调用。

③ 切换到 Edit Mode。

③ 选择 ‘if’ channel 并删除它。

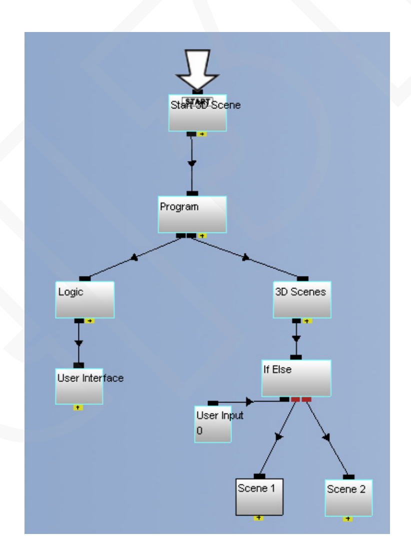
③ 断开 ‘Scene 2’ 和 ‘3D Scenes’ channels 之间的连接。

③ 拖动一个 IfElse channel 到 Channel 视图上并把它连接到 ‘3D Scenes’ channel 上。

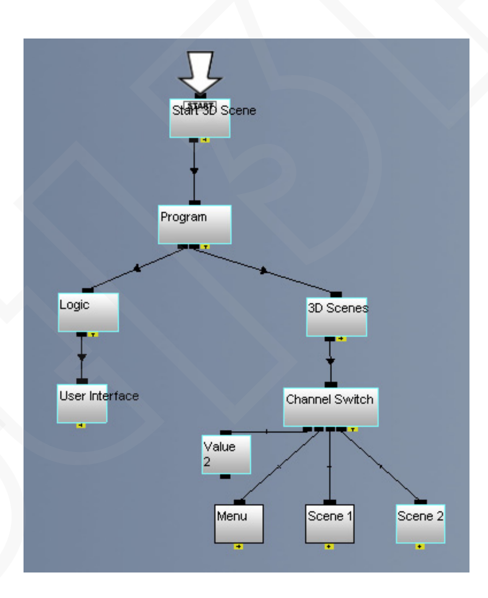
③ 连接 User Input channel 到 ‘If Else’ channel 的第一个子连接上。

③ 连接 ‘Scene 1’ channel 到 ‘If Else’ channel 的第二个子连接。注意 ‘Scene 1’ channel 没有被调用。

③ 连接 ‘Scene 2’ channel 到 ‘If Else’ channel 的第三个个子连接。注意 ‘Scene 2’ channel 被调用。当前的 channel 结构看上去与下图类似。



③ 切换到 Run Mode。
③ 按住‘1’键 注意‘Scene 1’channel 被调用并且‘Scene 2’channel 没有。
③ 释放‘1’键 注意‘Scene 1’channel 不再被调用并且‘Scene 2’channel 被调用。
③ 切换到 Edit Mode。
③ 添加一个 Trigger channel 并将它连接到‘User Interface’channel。
③ 拖动一个 User Input channel 到 Channel 视图上并把它连接到‘Trigger’channel 的第一个子连接上。
③ 添加一个 Channel Caller 并把它连接到‘Trigger’channel 的第二子连接上。 R 重命名 Channel Caller 为‘Event’。
③ 切换到 Run Mode。
③ 按空格键。 注意‘Event’channel 被调用一次。
③ 再次按空格键。 注意‘Event’channel 被再次调用。
③ 切换到 Edit Mode。
③ 选择并且删除连接到‘If Else’channel 的‘User Input’channel。
③ 选择‘If Else’channel 并删除它。
③ 拖动一个 Channel Switch channel 到 Channel 视图上并把它连接到‘3D Scenes’channel 上。
③ 双击‘Channel Switch’channel 打开它的属性窗口。 从列表中选择‘Channel Caller’并按‘Set’按钮。单击‘OK’按钮。
③ 添加一个 Value channel 并把它连接到‘Channel Switch’channel 的第一个子连接上。重命名该 Value 为‘Current Scene’。
③ 拖动一个 Channel Caller 到 Channel 视图上并把它连接到‘Channel Switch’channel 的第二子连接上。重命名该 Channel Caller 为‘Menu’。注意‘Menu’channel 现在被调用。
③ 连接‘Scene 1’channel 到‘Channel Switch’的第三个个子连接。
③ 连接‘Scene 2’channel 到‘Channel Switch’channel 的第四个子连接。
③ 改变‘Current Scene’的值为‘1’。注意‘Scene 1’channel 被调用。
③ 改变‘Current Scene’的值为‘2’。注意‘Scene 2’channel 被调用。



完成后的场景：

..\Tutorials\3.1 – Logic\Logic – Complete.cgr

3.2 数学

Values 是任何 Quest3D 项目的核心。例如它们被用来定义物体的颜色,位置和大小。数值还可以描述得分，程序设置，菜单选项等等。

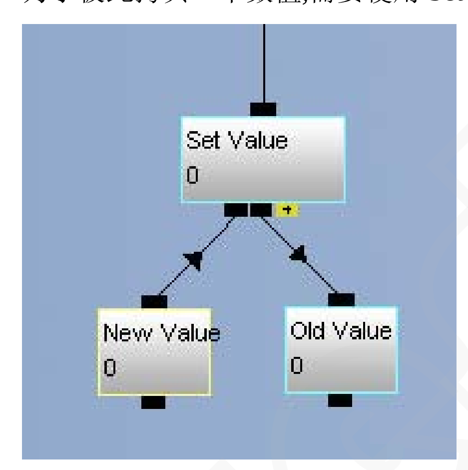
在 Quest3D 项目中数值是经常变化的。数学可用于计算新数值。

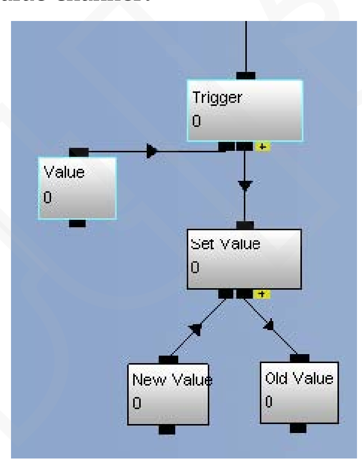
Set Value

下列声明描述了普通数学中的标准表示式：Old value = New value

在 Quest3D 中、上述声明必须如下转化：将 New value 复制到 New value

为了彼此拷贝一个数值,需要使用 Set Value channel。

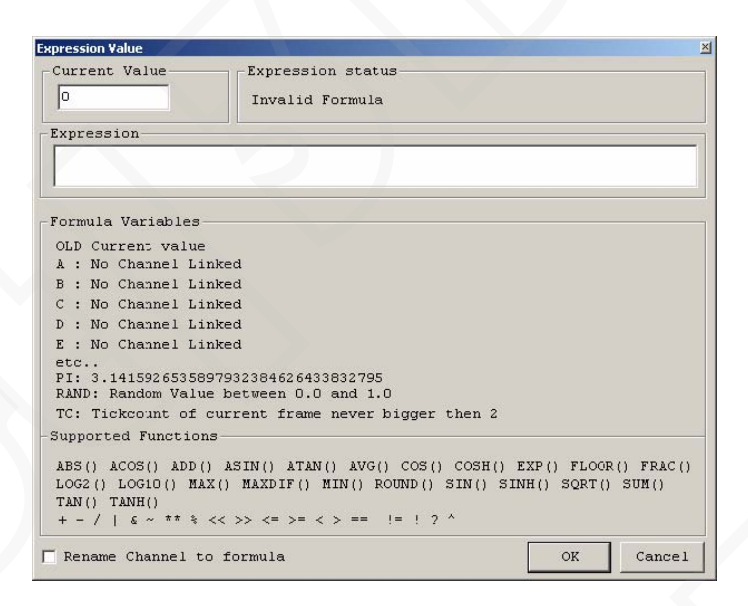




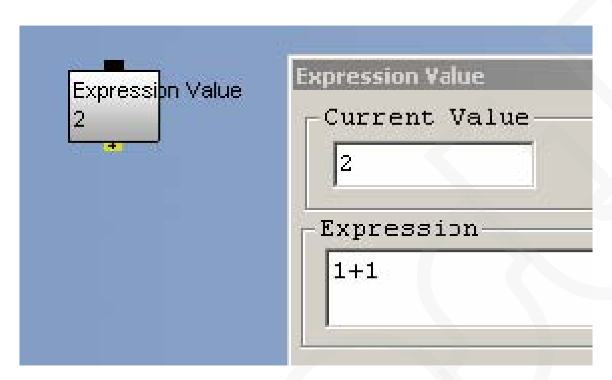
在实际的情况下 Set Value channels 仅需要执行一次。因此它们常常被 Trigger channel 调用。

Expression Value（表示式数值）

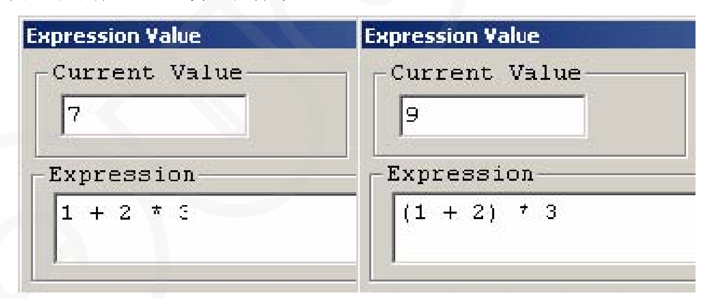
在 Quest3D 中，表达式可以用 Expression Value channel 描述。它起高级计算器的作用，它的属性窗口包含一个公式框和描述所有可用变量与算符的帮助文本。参考手册中详细解释了每一个要素。



Expression Value 的实际值等于 the channel 内部的表示式结果。该结果也显示在属性窗口的‘Current Value’中。

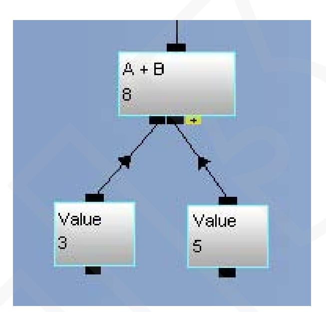


括弧（）可用于定义表达式的一部分相对另一部分的优先级。下图中的两个表达式不一样所以不会产生一样的结果。



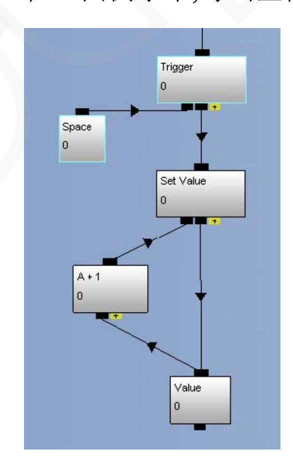
连接到 Expression Value 的 Value 也可作为 channel 内部表达式的一部分来使用。第一个子 channel 由字母‘A’代表、第二个由字母‘B’代表等等。

通过在属性窗口中选择‘Rename Channel formula’选项 Expression Value channel 可以它的表达式命名。

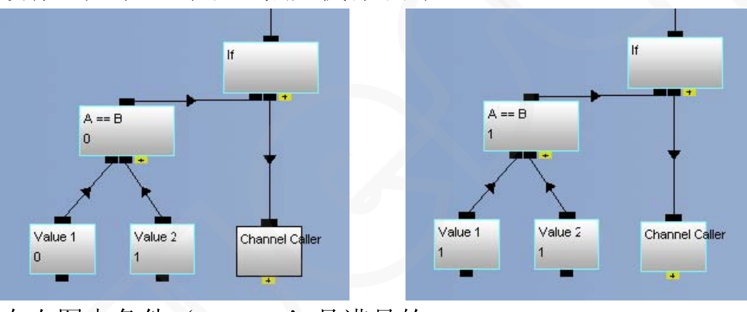


下列例子示范了常用的 Set Value, Expression Value 和 Value channel。

在上面例子中,每当空格键被按下,‘Value’ channel 值增加‘1’。

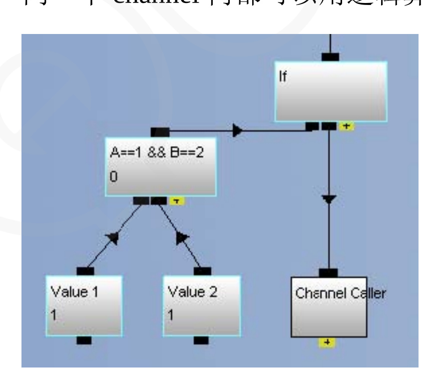


Expression Values 还可以作为 If 和 Trigger channels 的 input 输入条件。下图中，该结构应该读作‘如果 A 等于 B 然后执行下列 channels’。



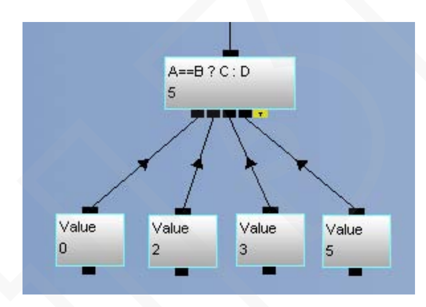
在右图中条件‘A = = B’是满足的。

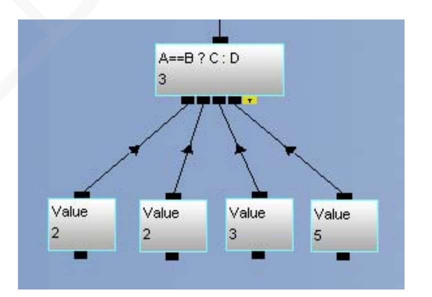
同一个 channel 内部可以用逻辑算子 AND (& &)与 and OR (||)连接多个表达式。



在上图中判断‘A = = 1 & & B = = 2’是否是满足的。该表达式应该读作、‘如果 A 等于 1 并且等于 2’。

表达式还可以用 If Else 声明的形式包含条件。下图中,表达式‘A = = B? C：D’应该读作‘如果 A 等于 B 该结果是 C 否则结果是 D’。





在右图里、条件‘A = = B’是满足的。该表达式的结果是‘C’。

Envelopes

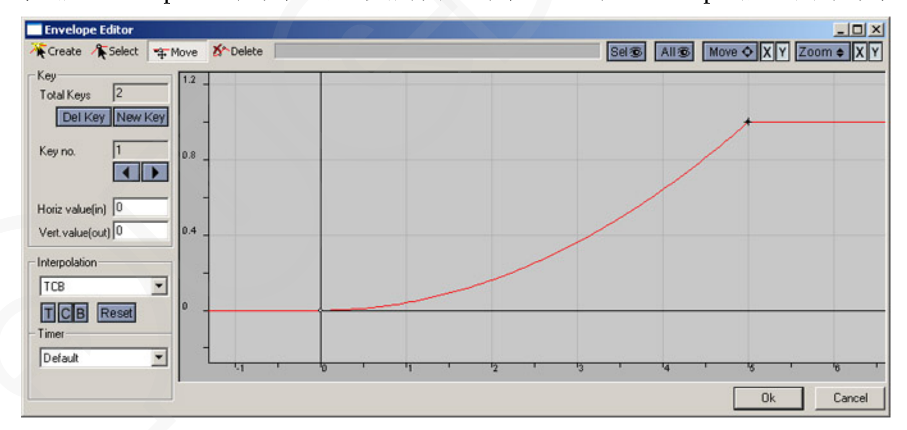
除在动画中扮演重要角色之外、Envelopes 也有其他的用途。

Envelope channel 中的曲线图是一个二元函数。对于一些高级仿真和模型，实际的公式可能会插入很多坐标对到该曲线图里。Quest3D 将在这些点之间运用内插法。



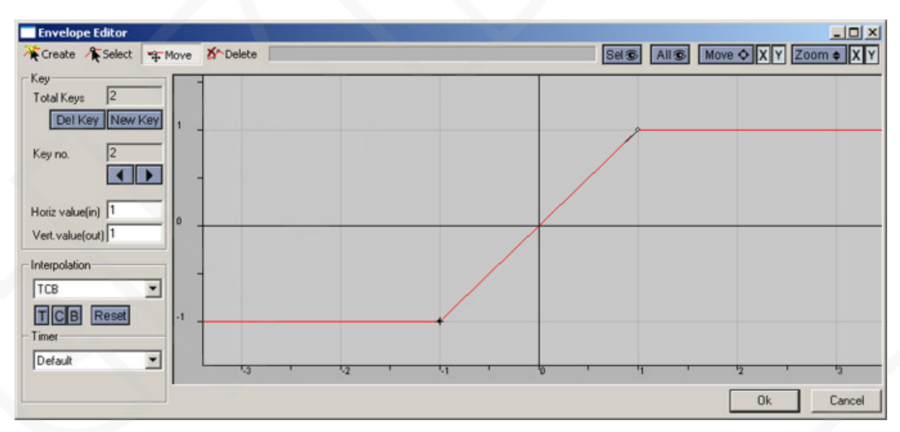
上图中的 envelope 曲线图表示公式‘y = x’。

依靠 Envelope 一个数值可以变为另一个值。一个 Envelope 描述两个数值之间的关系。



角色 motionset blending 可以以行走的速度为基础。上述曲线图将‘0’到‘5.0’之间的速度信号输入数值换算为在‘0’和‘1.0’之间的 blending 输出值。该曲线图是非线性的。

Envelopes 还可以限定值的范围。



上述曲线图在-1 到 1 之间限制输入输出值。

实例

数学是许多 quest3d 程序的一个重要方面。该实例练习使用下面的 channel：Value Set Value Expression Value 和 Envelope。

需要 quest3d 场景：

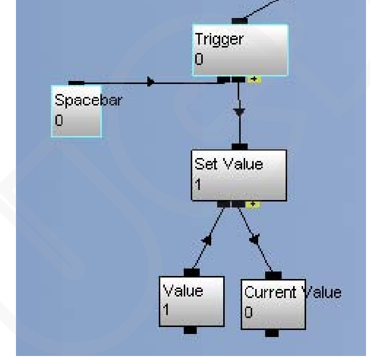
③ ..\tutorials\3.2 math\math.cgr

需要的模板：

③ Variables \ Value \ Set Value
③ Variables \ Value \ Expression Value
③ Variables \ Value \ Value

Step by step

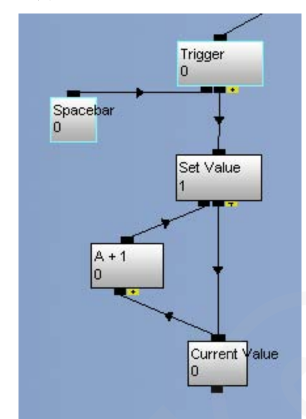
③ 打开‘Math.cgr’场景。它包含一个简单的逻辑结构。
③ 拖动一个 Set Value channel 到 Channel 视图上并把它连接到 Trigger 第二子连接上。
③ 添加一个 Value channel 并把它连接到‘Set Value’channel 的第一个子连接上。
③ 改变这个 Value channel 的值为‘1’。
③ 拖动另一个 Value channel 到 Channel 视图上并把它连接到‘Set Value’channel 的第二子连接上。重命名该 Value 为‘Current Scene’。



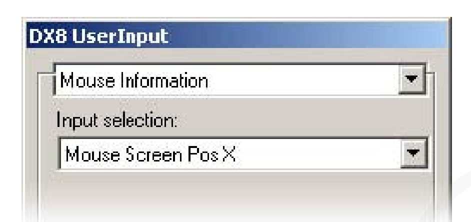
③ 切换到 Run Mode。

③ 按空格键。注意该‘Set Value’channel 被触发一次。同时注意第一个 Value channel 的值被拷贝到第二个 Value channel 中。现在两个 Value Channel 都是‘1’。
③ 切换到 Edit Mode。

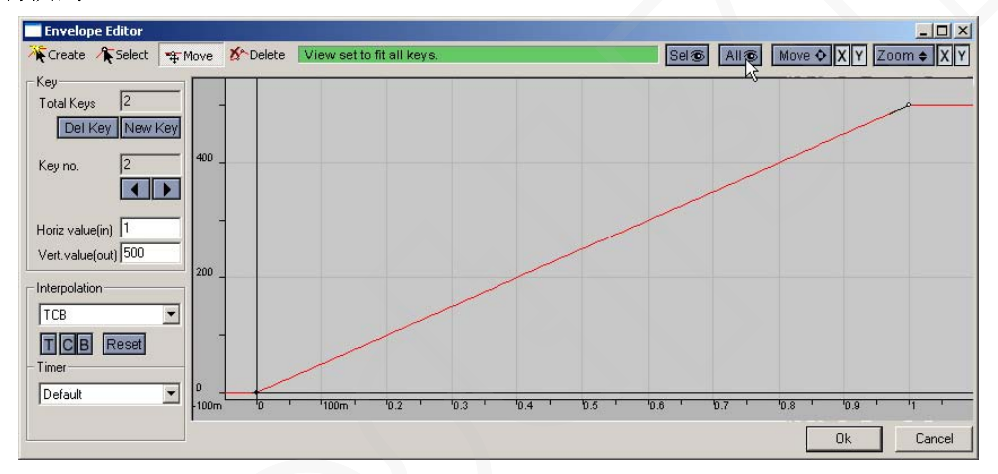
③ 添加一个 Expression Value channel 并将它连接到‘Calculations’channel。
③ 双击‘Expression Value’channel 打开它的属性窗口。
③ 在‘Expression’栏键入’1’。注意现在‘Current Value’等于‘1’。同时注意，‘Expression Value’channel 的值是‘1’。
③ 单击‘OK’按钮。
③ 拖动一个 Value channel 到 Channel 视图上并改变它的数值为‘3’。将它连接到‘Expression Value’channel。
③ 双击‘Expression Value’channel 打开它的属性窗口。
③ 在‘Expression’栏键入‘A’，引用子连接的值。注意现在‘Current Value’等于‘3’。同时注意，‘Expression Value’channel 的值是‘3’。
③ 选择并删除连接到‘Set Value’channel 的第一个子连接的 Value channel。
③ 添加另一个 Expression Value channel 并把它连接到‘Set Value’channel 的第一个子连接上。
③ 连接‘Current Value’channel 到‘Expression Value’channel。
③ 双击‘Expression Value’channel 打开它的属性窗口。在‘Expression’栏输入下列公式:‘A + 1’。
③ 选择‘Rename Channel to formula’选项并按‘OK’。



③ 切换到Run Mode。
③ 按空格键。注意‘Current Value’channel的值增加了1 (‘1 + 1 = 2’)
③ 再次按空格键。注意该数值再次增加(‘2 + 1 = 3’)。
③ 切换到Edit Mode。
③ 改变该公式Expression Value为‘A < 4? A+1 : 0’。注意该公式应该读作‘如果A小于4，结果是A加1，否则结果是0。
③ 切换到Run Mode。
③ 按空格键。注意现在‘Current Value’是 4 。
③ 再次按空格键。注意现在‘Current Value’是0。
③ 添加一个User Input channel将它连接到‘Calculations’channel。
③ 双击这个‘User Input’channel打开它的属性窗口。从第一个下拉列表中选择‘Mouse Information。从第二下拉列表中选择‘Mouse Screen Pos X’。单击‘OK’按钮。



③ 切换到Run Mode。
③ 从左至右移动鼠标。注意‘User Input’channel值的改变。该最大值取决于显示器的分辨率。
③ 切换到Edit Mode。
③ 选择并且删除‘User Input’和‘Calculations’channel之间的连接。
③ 拖动一个Envelope channel到Channel视图上并把它连接到‘Calculations’channel上。
③ 连接该‘User Input’channel到‘Envelope’channel。
③ 双击该‘Envelope’channel打开的它的属性窗口。在(0, 0) 和(500, 1)处创建键。 在‘All’按钮上单击以查看全部的曲线图。
③ 单击‘OK’按钮。
③ 切换到Run Mode。
③ 从左至右移动鼠标。注意‘Envelope’channel的值在‘0’和‘1.0’之间变化。
③ 切换到Edit Mode。



完成后的场景：

③ ..\tutorials\3.2 math\math complete.cgr

3.3 For Loop

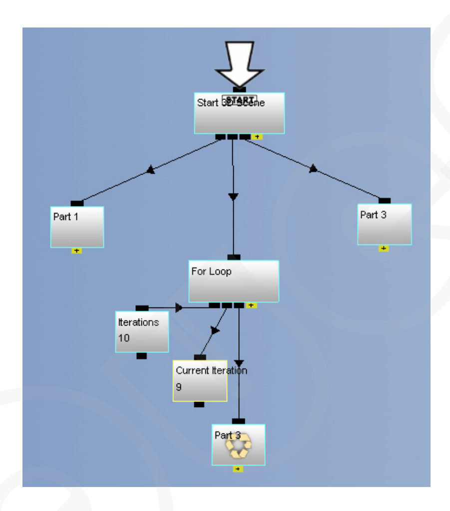
高级实时交互场景可能由数以百计的单元组成。如果不考虑效率，某一场景可能对显卡要求非常苛刻。项目管理可以摆脱这种情况。没有恰当的结构，管理如此多的东西可能变得复杂而不能处理。

在场景中的多个对象彼此共用一个或多个属性。在该项目的设计阶段，建议考虑将对象组分割。

例如：考虑允许用户装饰他或她的虚拟家的应用程序。可用的各种型式的家具可能看上去互不相同，除了它们都是全部三维模型。此外，该用户也许要移动，旋转和缩放这些家具并把它们放入该虚拟场景。这些属性被家具组共用。

For Loop

如同许多其他的编程语言一样，Quest3d 特色之一是所谓的‘For Loop’功能。每帧中 For Loop 会将程序的特定部分执行多次。只有当 For Loop 的所有‘iterations’已经全部执行完后该程序才会继续下一部分。

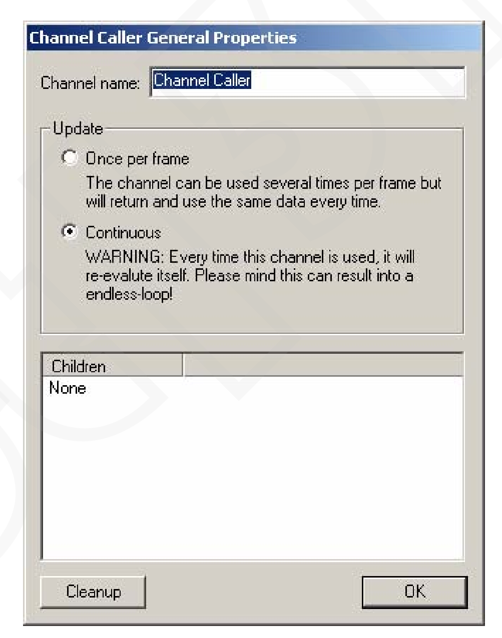


在上图中，‘Part 1’是第一个被处理的。接下来，‘Part 2’通过 For Loop 功能执行 10 次。最后，该程序进行到‘Part 3’。

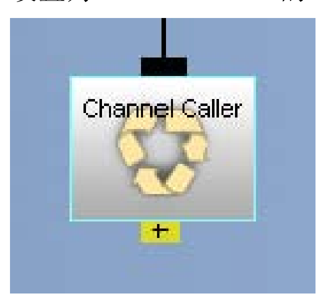
驾驶仿真可能要求五十汽车在同一场景。使用一个 For Loop 允许你创建单个汽车对象并渲染五十次数，甚至运用不同的位置，形状，颜色，而不是使用五十个不同的汽车对象。

每帧更新

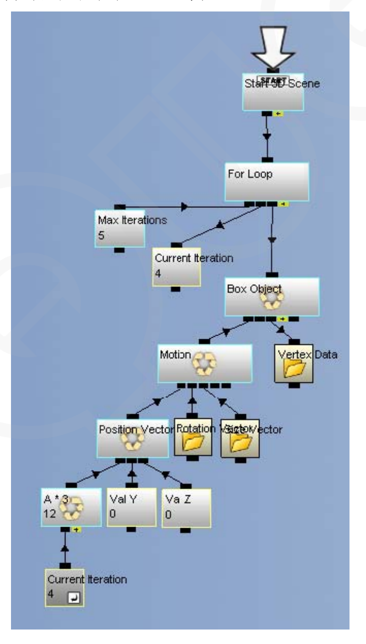
Quest3d 通常每一帧仅更新一次 channel。这是出于性能和安全性考虑。而 For Loop 中的 Channel 每一帧需要更新多次。这种 channels 的更新设置需要设置为'Continuous '。该‘Continuous’更新选项位于各个 channel 的‘General Properties’窗口中。

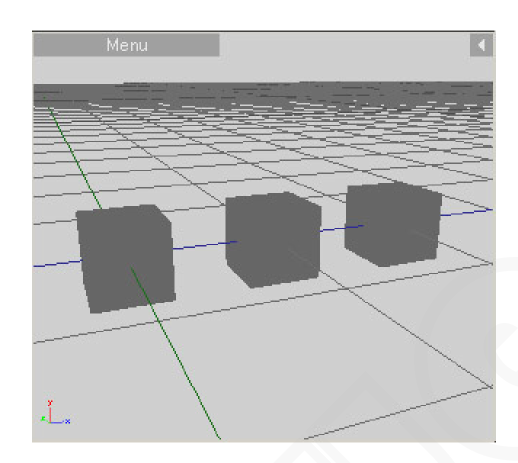


设置为‘Continuous’的 Channel 可以通过一个标记来辨认。



一个 For Loop channel 的‘Max Iterations’子 channel 定义了子程序执行的总次数。在 ForLoop 中‘Current Iteration’Value 被用于在各个实例中创建变化的量。简单例子是每个对象实例的不同 X 坐标。



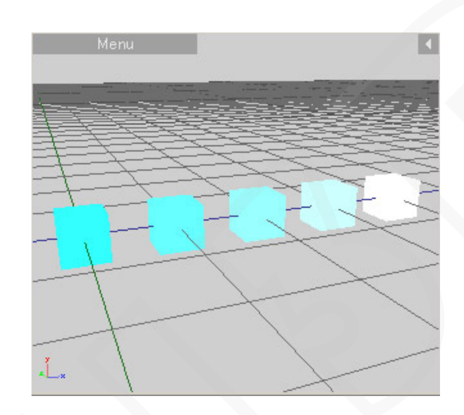


Expressions（表达式）

在 Current Iteration 和 Max Iterations Values 之间的关系也能被使用。下图中，Expression

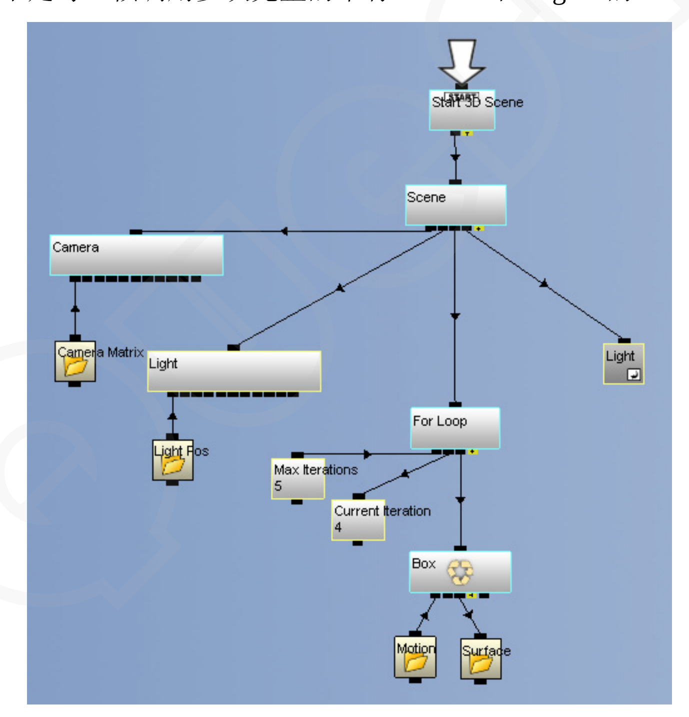


Value‘(A+1) / B’产生一个从‘0.2’到‘1.0’的递减。



Rendering scenes（渲染场景）

保持一个 For Loop 循环中的 channel 数量尽可能小是个好习惯。最好仅循环对象自己，而不是每一帧调用多次完整的带有 Camera 和 Lights 的 Render。



在上图中，在该 For Loop 之前，Camera 和 Light 仅仅被调用一次。基于‘Current Iteration’值的对象被调用多次。最后，Light 被再调用一次以关闭它，所以该 Light 不会影响其他物体。

Nature painting 与 For Loops

如同在 2.9 章讨论的一样，Nature Painting 也是为在场景中渲染大量三维物体而设计的。由于使用了硬件优化 Nature Painting 的渲染速度是非常快的，特别是特定的静态元素。 动态对象应该总是使用 For Loop 渲染。

实例

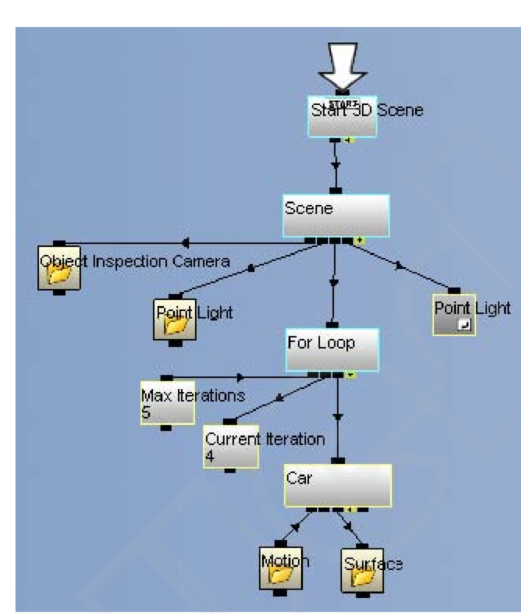
本实例使用一个 For Loop 结构在屏幕上渲染多个汽车。

需要的模板：

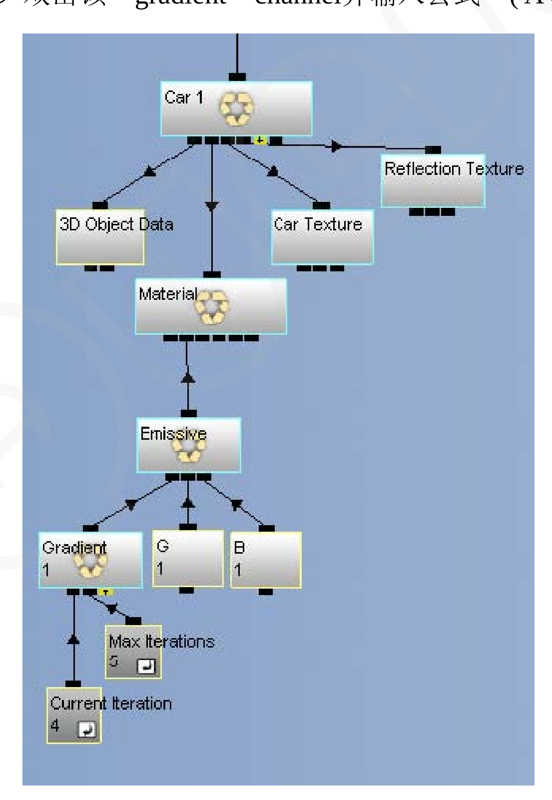
- 3D Items \ Start 3D Scene
- Logic \ Channel Caller
- Scene \ Cameras \ Object Inspection Camera
- Scene \ Light \ Point Light
- Logic \ For Loop
- 3D Models \ Car
- Variables \ Value \ Expression Value (x2)

Step by step

- 新建一个项目。
- 拖动一个Start 3D Scene到Channel视图上。右键单击‘Start 3D Scene’并选择‘Set as start channel’。
- 添加一个Channel Caller并重命名为‘Scene’。 将它连接到‘Start 3D Scene’channel。
- 拖动一个Object Inspection Camera模板到Channel视图上并把它连接到‘Scene’channel上。
- 通过点击' Groups '标签关闭Animation 3D View窗口。
- 添加一个Point Light模板并将它连接到‘Scene’ channel。
- 拖动一个ForLoop模板到Channel视图并将它连接到' Scene ' channel。
- 改变‘Max Iterations’channel的值为‘5’。
- 添加一个Car模板并将它连接到For Loop channel。
- 创建一个Point Light channel的快捷方式并将它连接到‘Scene’channel。在渲染之后light将关闭所以它不会影响其他的render。
- 单击'Animation 3D View标签以查看现在的场景。 注意仅有一个汽车是可见的。



③ 删除car的‘Position’Vector中的‘Val X’channel。
③ 拖动一个Expression Value到该模板上并连接它到‘Position’channel的第一个(空闲的)子连接上。
③ 创建一个‘Current Iteration’channel的快捷方式并将它连接到‘Expression Value’channel。
③ 双击该‘expression value’channel并输入公式，‘A * 3’。 选择‘rename Channel to formula '选项单击' OK '。
③ 在‘A * 3’channel上按右键并选择‘general properties’。在该属性窗口中，选择‘continuous’选项。 现在该channel将在每一帧中更新多次。单击‘OK’按钮。
③ 设置‘Position’,‘Motion’和‘Car’channel为‘Continuous’。 注意在For Loop中所有这些channel都被作为每个实例需要被更新。 在上次设置'continuous'之后，五个汽车会显示在Animation 3D View窗口中。
③ 切换到Run Mode。
③ 环绕汽车以查看它们。
③ 切换到编辑方式。
③ 从car的' Car 1 '表面，删除' Emissive ' Vector的' R ' Value。
③ 添加一Expression Value并重命名为‘Gradient’ 将它连接到‘Emissive’channel的第一个(空闲的)子连接上。 注意汽车颜色的改变。
③ 创建一个‘Current Iteration’channel的快捷方式并将它连接到‘Gradient’channel。
③ 创建一个‘Max Iterations’channel的快捷方式并将它连接到‘Gradient’channel的第二子连接上。
③ 设置上面的‘Gradient’channel和‘Emissive’和‘Material’channel为‘Continuous’。
③ 双击该‘gradient’channel并输入公式‘(A+1) / B’。



③ 单击' OK '。 注意汽车颜色的改变，从青绿色到淡灰。

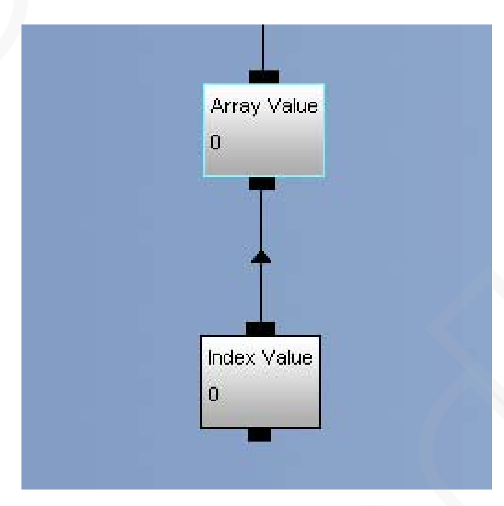
完成后的场景：

③ ..\Tutorials\3.2 – Advanced logic\Advanced logic – Complete.cgr

3.4 数组

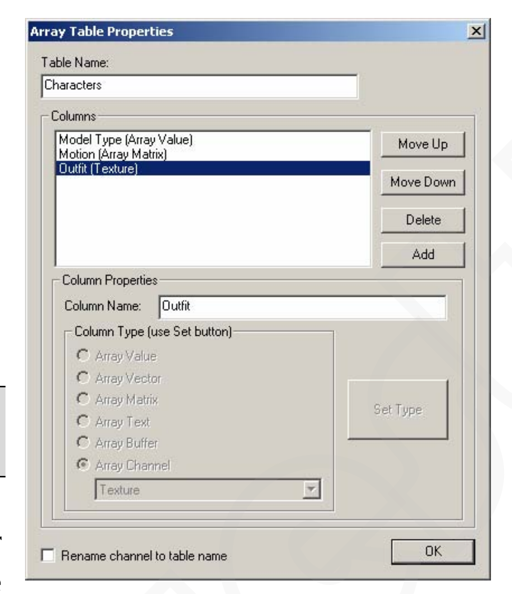
变量可以存储特定类型的信息。Quest3D 中最常用变量类型是 value, vector, matrix 和 text。它们都可以包含对应类型的单个实例。例如 Value channel 可以存储一个数值。

数组可以存储特定类型的多个实例。在数组内部每个实例或记录都有一个固定的位置，并可使用索引值存取。



Tables

在 Quest3D 中数组是 Table 的一部分。项目初始化时 Array Table channel 仅需调用一次来初始化。在 Quest3D 程序中连续调用 Array Table 没有副作用——它仍然仅初始化一次。



Array Table channel 的属性窗口允许添加并定义列。

栏目名可以在任何时候改变——它们仅被用作标签。

> 同名的数组不结合。始终不要重名以避免该问题。

在该 Array Table 的属性窗口和 Array Manager 窗口(在本章稍后描述)中 'Move Up '和 'Move Down ' 按钮影响列表中列的次序。

Data types（数据类型）

数组列可以储存多种类型的数据。除了上述的变量类型(value, vector, matrix 和 text),在

Quest3D 中创建一系列任意 channel 也是可能的。 通过从‘Array Channel’下拉列表选择一个类型来定义它们。

最后一类例子包含三维物体或纹理的数组。 高级的使用可以是一个 Network Values 数组。

注意列类型必须通过单击按钮提交。

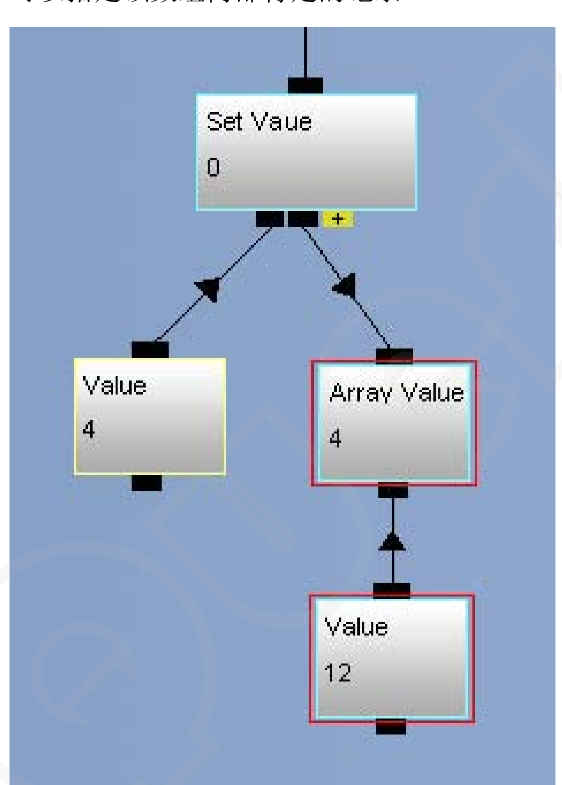
Data storage and references（数据存储和引用）

数据保存在 Array Table channel。为了在 Quest3D 项目中存取和使用数据，需要使用引用。存在多种类型的引用 channel，每一个都有列类型。例如一个 Array Value channel 是对那个数据类型优化过的。

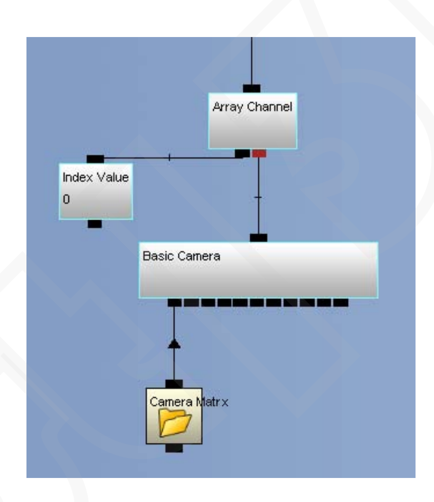
一些 Array channel 的属性窗口用于指定引用应该存取哪个 table 和列。注意 Array Channel 的列表可以仅设置一次，因为它的 Channel Switch。

Setting Arrays（设置数组）

为了在一个数组中存储数值，应使用适当类型的‘Set’operator。通过连接一个索引值，可以指定该数组内部特定的记录。

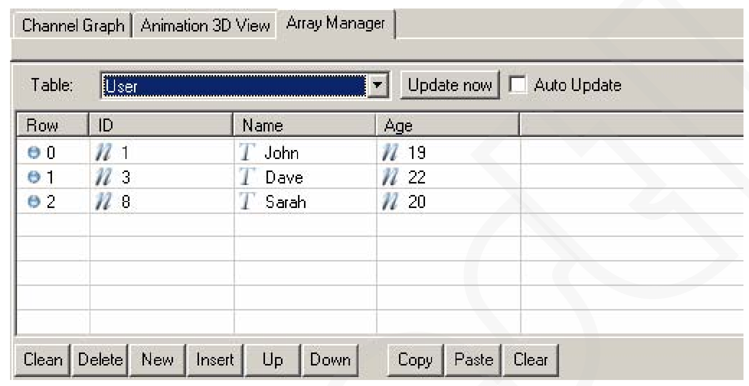


注意该 Array Channel 有两个子连接块。第一个是索引数值。第二个可用于连接一个适当的类型的 channel。 它的子将被传递到 Array Channel。



Array Manager

在 Quest3D 中 Array Manager 显示包含在项目中的所有数组，每一个表。它可以通过单击 Channel 和 Animation Section 中的‘array manager’标签，或通过切换到 Array Section 来存取。



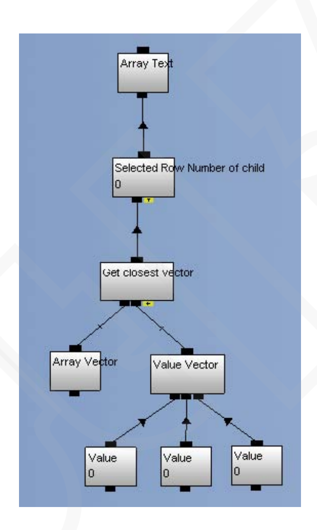
Array Manage 中的一栏称为记录或一个字段。值类型允许直接输入。正文类型也可以直接(在‘T’图标上单击一次)编辑，或者通过属性窗口(双击‘T’图标)编辑。

如果可用，单击其他字段将打开该类型的属性窗口。

在该 Array Table channel 的属性菜单中可以改变列的次序。单击'Update Once '按钮将刷新所有已显示的列。启用‘Auto Update’选项将持续刷新。

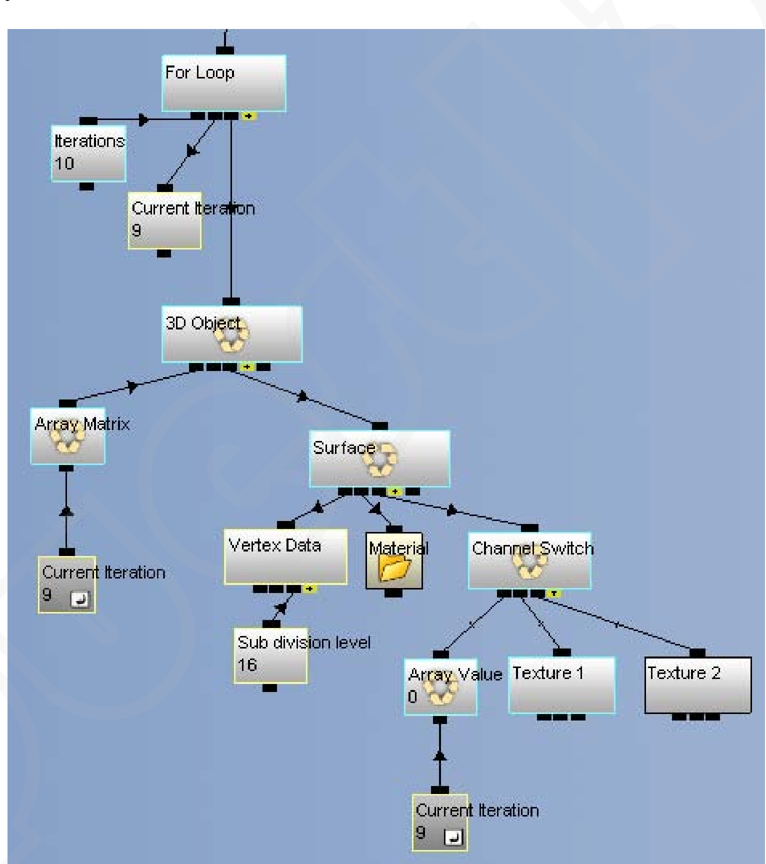
Commands, operators and info

Array Command 用来中数组中操作记录。各种 Array Operator 和 Info channel 可用于检索信息。下图显示一个高级用法。



For Loops

For Loop 和 Array channel 组合使用能实现非常强大的功能。‘current iteration’ Value 用于索引该数组。



参见 3.3 章得到更多有关 For Loops 的信息。

实例

数组可以包含大量数据。本实例说明在一个数组中数据是如何存取的。

需要 quest3d 场景：

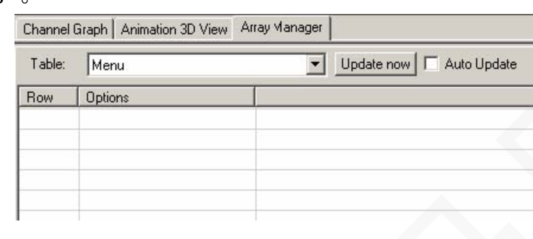
③ ..\Tutorials\3.4 – Arrays\Arrays 1.cgr

需要的模板：

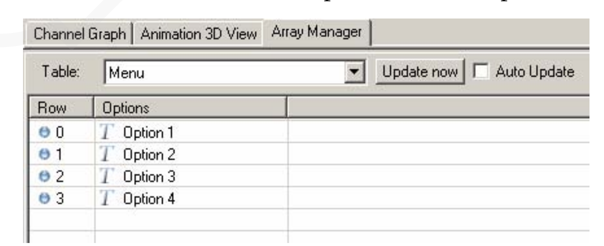
③ Variables \ Arrays \ Array Value
③ Variables \ Value \ Value

Step by step:

③ 打开‘Math.cgr’场景。 它包含一个文本对象。
③ 确定该场景使用Project Camera显示在Animation 3D View窗口中。
③ 拖动一个Array Table到Channel视图上。 并把它与‘Main’channel连接在一起。 双击Array Table channel打开的它的属性窗口。
③ 重命名该table为‘menu’。
③ 通过单击‘Add’按钮创建一个新列。
③ 选择该列并重命名它为‘options’。
③ 选择‘array text’单选按钮并单击‘Set’按钮确认。
③ 选择‘Rename channel to table name’选项并按‘OK’。
③ 添加一Array Text channel。 双击它打开它的属性窗口。
③ 从‘table’下拉列表中，选择‘menu’。
③ 从‘Column’下拉列表中，选择‘Options’。
③ 选择‘Update channel name’选项并按‘OK’。现在该channel被设置为一个正确的表和列的引用。



③ 单击该“array manager’标签。注意存在一个名为‘Menu’的表，包括一列‘options’。
③ 在‘Options’列的第一栏(行‘0’)，输入‘Option 1’。
③ 单击‘New’按钮创建一个新记录。 T在列‘1’中输入‘Option 2’。
③ 添加更多记录，名为“Option 3’和‘option 4’。



③ 单击‘Channel Graph’标签。
③ 将‘menu : Options ' channel连接到2D Text from Texture ' channel。注意在Animation 3D View窗口中文本变换成‘Option 1’。

- ③ 添加一个Value并将它连接到‘Menu : Options’channel。重命名为‘menu index’。
- ③ 改变这个‘Menu Index’channel的值为‘1’。注意在Animation 3D View窗口中文本变换成‘Option 2’。
- ③ 通过改变‘Menu Index’channel的值为‘1’测试另一个选项。

完成后的场景：

- ③ ..\Tutorials\3.4 – Arrays\Arrays 1 – Complete.cgr

你可以保存现在的 channelgroup 到你的项目目录。 下述步骤处理一个新的场景。

需要 quest3d 场景：

- ③ ..\Tutorials\3.4 – Arrays\Arrays 2.cgr

需要的模板：

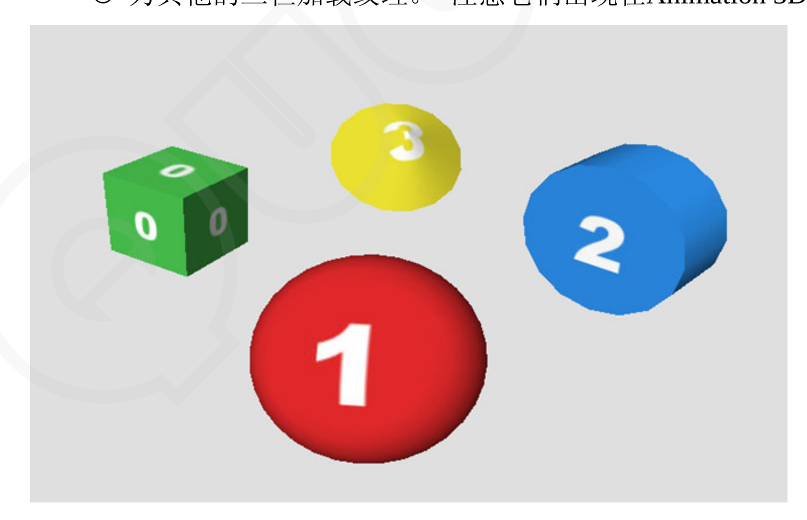
- ③ Variables \ Value \ Array Value
- ③ Variables \ Matrix \ Set Matrix
- ③ 3D Items \ Null Object
- ③ Variables \ Matrix \ Array Matrix

Step by step:

- ③ 打开‘Arrays 2.cgr’场景。 它包含一些基本对象和一个配置了部分的表。
- ③ 拖动一个Array Value到Channel视图上。双击它打开属性窗口。
- ③ 从‘table’下拉列表中，选择‘Objects’。
- ③ 从‘Column’下拉列表中，选择‘’。
- ③ 选择‘Update channel name’选项并按‘OK’。
- ③ 连接‘Objects : Type’channel到‘Object Type Switch’channel的第一个子连接。
- ③ 创建一个‘Current Iteration’channel的快捷方式并将它连接到‘Objects : Type’channel。
- ③ 在‘Object : Type’channel上按右键并选择‘general properties’。 在该属性窗口中，选择‘continuous’更新选项。 单击‘OK’按钮。
- ③ 确定Animation 3D View窗口是打开的并使用Project Camera显示该场景。
- ③ 单击‘array manager’标签。 注意在‘Objects’表中的两列，‘Motion’和‘Type’。
- ③ 在‘Object Type’列中的第一栏（行0）键入‘1’。 注意在Animation 3D View窗口中的基本对象改变了形状。
- ③ 在‘Type’列中的第二栏（行1）键入‘2’。 注意在Animation 3D View窗口一个对象改变了形状。
- ③ 填写其他的两个记录。 (注意：‘0’= Box, ‘1’= Sphere, ‘2’= Cylinder, ‘3’= Cone)
- ③ 单击‘Channel Graph’标签。
- ③ 拖动一个Array Matrix到Channel视图上并把它连接到Set Matrix channel上。 双击它打开它的属性窗口。
- ③ 从‘table’下拉列表中，选择‘Objects’。
- ③ 从‘Column’下拉列表中，选择‘Motion’。
- ③ 选择‘Update channel name’选项并按‘OK’。
- ③ 添加一个Value并将它连接到你刚才创建的‘Objects : Motion’Array channel。重命

名该Value为‘Current Object’。

③ 进入Animation Section。
③ 在列表中单击‘Object Mover’物体选择它
③ 来回移动‘object mover’。 注意一个简单对象随着它移动。
③ 进入Channel Section。
③ 改变‘Current Value’的为‘1’。
③ 进入Animation Section。
③ 在列表中单击‘Object Mover’物体选择它
③ 来回移动‘object mover’。 注意与索引‘1’对应的简单对象随着它移动。
③ 进入Channel Section。
③ 双击该‘Objects Table’channel打开的它的属性窗口。
③ 通过单击‘Add’按钮创建一个新列。 命名为‘texture’。
③ 选择该‘array channel’单选按钮并从下拉列表中选择‘texture’。 单击‘set’按钮确认。 P单击‘OK’按钮。
③ 拖动一个Array Channel到Channel视图上并连接它到‘Box’对象‘Surface’channel的正确的子连接上。
③ 双击该‘Array Channel’打开的它的属性窗口。 配置它为一个在Objects表中的一个Texture列引用。 S选择‘Update channel name’选项并按‘OK’。
③ 创建一个‘Current Iteration’channel的快捷方式并将它连接到‘Objects : Texture’channel。
③ 创建三个‘Objects : Texture’channel的快捷方式并分别将它们连接到‘Sphere’，‘Cylinder’和‘Cone’对象的‘Surface’channel的正确的子连接块上。
③ 单击the Array Manager标签。
③ 在‘Objects’表‘Texture’列上，双击第一栏打开的该记录的属性窗口。 定位并加载一个纹理。 注意在Animation 3D View窗口中，第一个对象（‘0’）现在有一个纹理。
③ 为其他的三栏加载纹理。 注意它们出现在Animation 3D View窗口中。



完成后的场景：

③ ..\Tutorials\3.4 – Arrays\Arrays 2 – Complete.cgr

3.5 Multiple channel groups

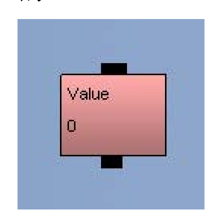
高级的 Quest3D 程序可能是复杂的项目。在某些特定的情况下可以将一个程序分割为多个部分

4.1 章项目管理讨论使用多 channelgroups 工作的好处。本章解释它在 Quest3D 中确切的作用。

使用特定的 channel：‘Public Channels’和‘Public Call Channels’可以将多个 channel 连结在一起。

Public Channels

Public Channel 是可以从外部 channel groups 调用的 channel。在 QueSst3D 中他们是红色的。



Public Call Channels

Public Call Channels 连接到 Public Callers。在 QueSst3D 中他们是蓝色。

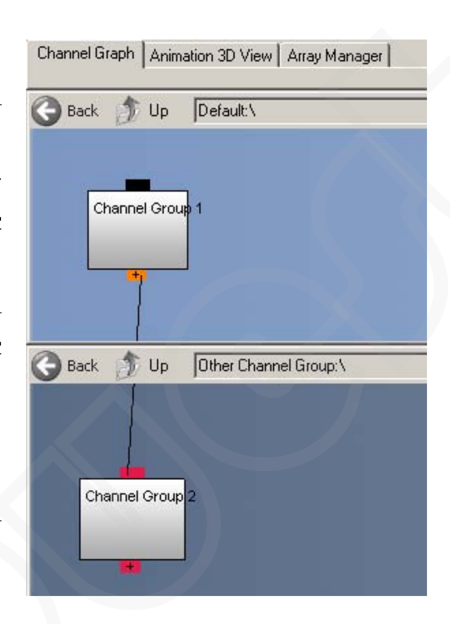
任何基类型的 Channel 都可以转化为 Public Callers。最常用类型是：Channel Caller, Value, Vector, Matrix, 3D Object, Camera 和 Light。尽可能使用这些类型的 Public Callers 以实现易用和内存管理的目的。

Linking between channel groups

连接一个 channel 与另一个 channel group 中的 channel 将自动转换它为 Public Channel。一个 Public Call Channel 在另一个 channel group 中被创建。

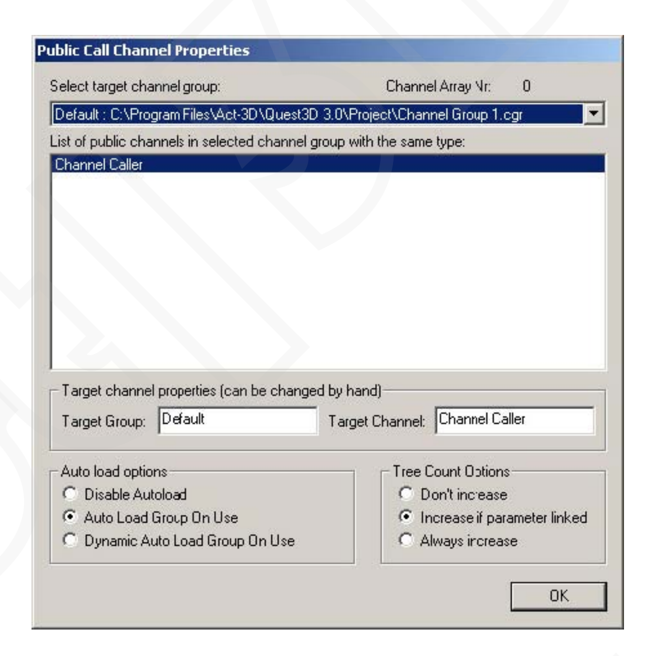
从上下文菜单中选择‘Public Channel Functions > Make Channel Public’可以手动地把 Channels 变为 Public Callers。或者，单击‘P’键。

上下文菜单也提供一个选项手动地创建 Public Call Channels：‘Public Channel Functions > Convert to Public Call Channel’。



Public Call Channel properties

通过 channel 的属性窗口可以指定 Public Channel (相同的 base 类型)到 Public Call Channel 的连接。

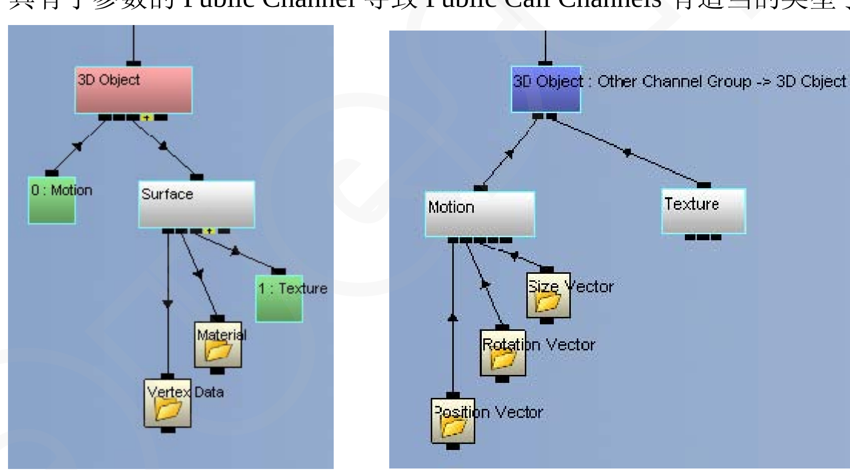


Functions

函数是可以使用参数调用的子程序。一种 Public Caller，在 Quest3D 中它们也是红色的。参数被用作函数的输入数据。在 Quest3D 中他们是绿色的。

从上下文菜单中选择‘Public Channel Functions > Convert to Parameter Channel’可以手动地把 Channel 转换成参数。

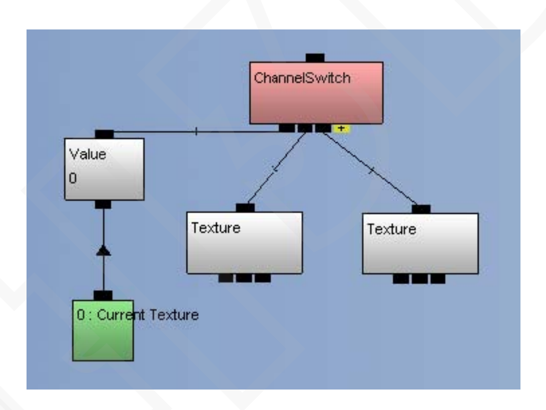
具有子参数的 Public Channel 导致 Public Call Channels 有适当的类型子连接。



函数可以在每次需要时被调用。可以为每个实例提供不同的参数，从而产生不同的效果。

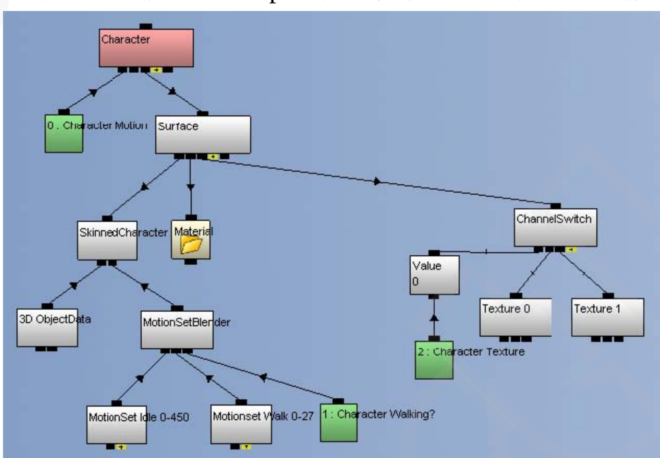
Channel Switches

在‘Selector’子连接可以作为参数被连接之前，Channel Switch 类型的函数必须包含一个额外的 Value channel。这是 Channel Switch 的特性。

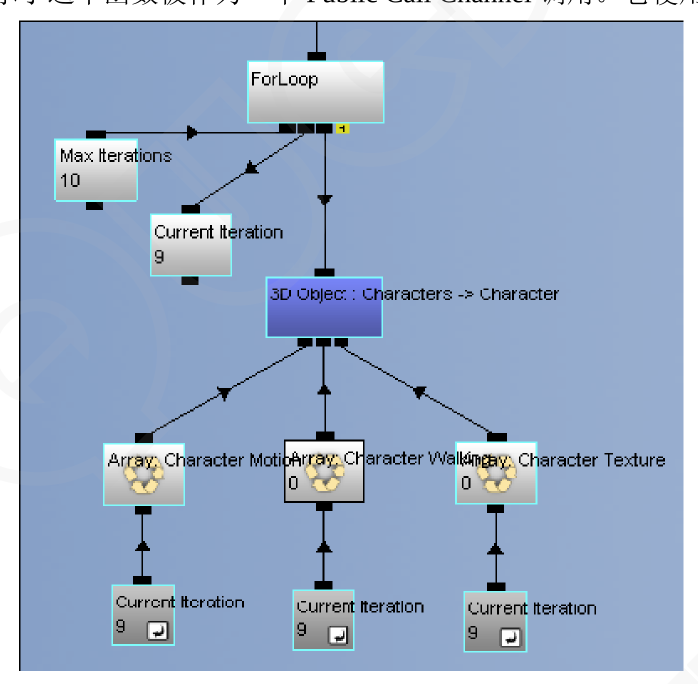


For Loops

函数也可能在 For Loop 结构之内被调用。数组可以作为参数连接以实现更强大的功能。



上图显示了一个带有参数的动画角色，参数作为它的 motion,当前的动作和纹理。 下图显示了这个函数被作为一个 Public Call Channel 调用。它使用 For Loop 中的数组作为参数。



实例

需要 quest3d 场景：

③ ..\Tutorials\3.5 – Multiple channel groups\Multiple channel groups.cgr

③ ..\Tutorials\3.5 – Multiple channel groups\3D scene render.cgr

③ ..\Tutorials\3.5 – Multiple channel groups\Car.cgr

需要的模板：

③ Variables \ Value \ Value

Step by step:

③ 打开场景‘Multiple channel groups.cgr’。 I它包含一个简单的项目结构。

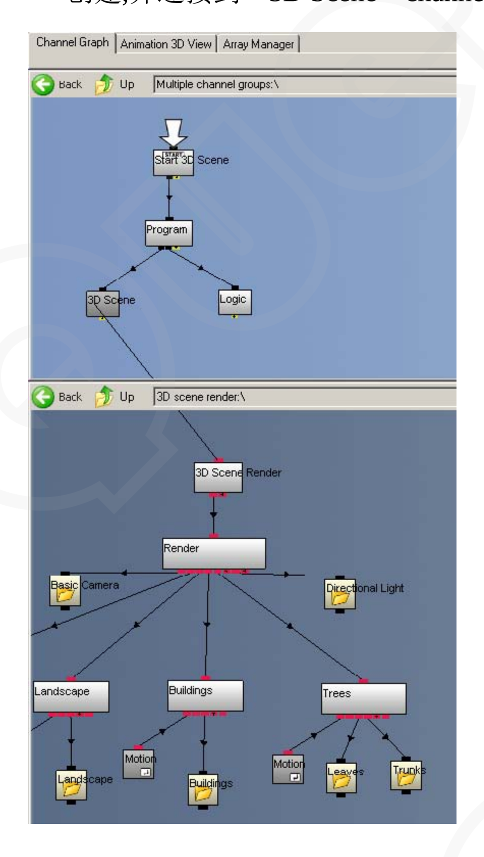
③ 从这应用程序菜单中，选择‘File > Import...’。 在上述指定的目录中找到‘3D scene render.cgr’并单击‘Open’。使用‘3D scene render’作为pool名称。该文件存储一个简单场景包含Render, Camera和Light channel。

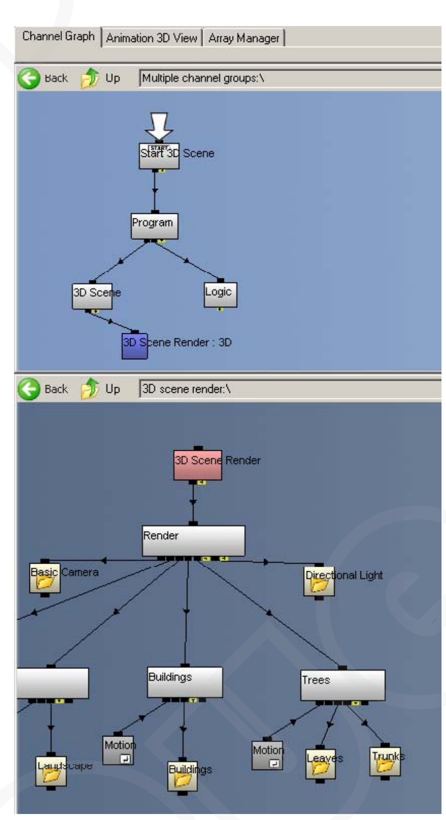
③ 在‘groups’标签上单击。

③ 从‘groups’标签中，拖动pool名称‘Multiple channel groups’到Channel视图上。

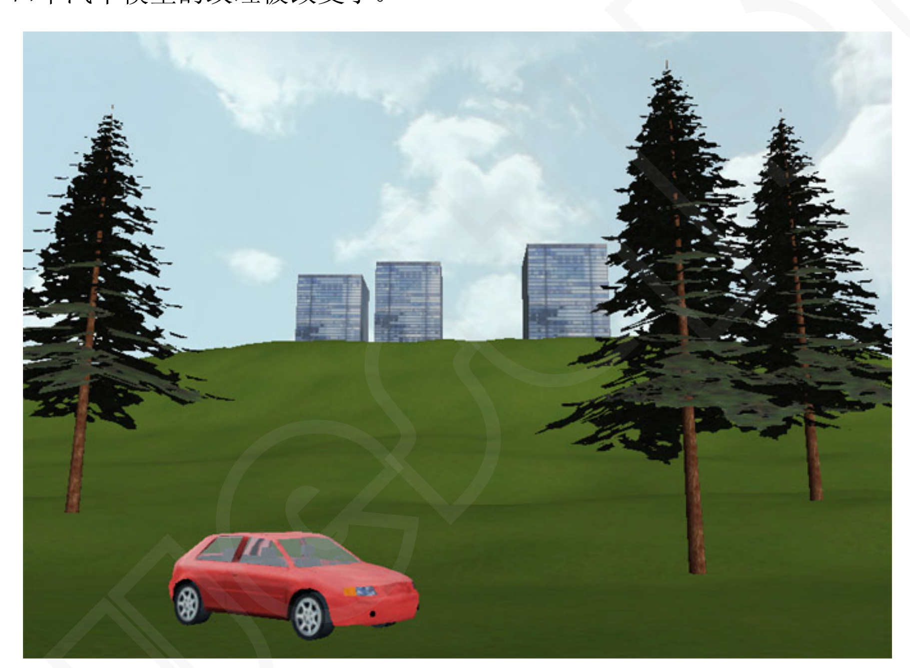
③ 从‘groups’标签中，拖动pool名称‘3D scene render’到Channel视图的下部。

③ 连接Channel视图下方的‘3D Scene Render’channel到Channel视图上方的‘3D Scene’channel。注意在Channel视图下方的‘3D Scene Render’channel变为红色,并且在其下面的channels被调用(淡蓝色轮廓)。 注意在Channel视图上方的一个蓝色channel被创建,并连接到‘3D Scene’channel。





③ 从上述指定的目录中导入文件‘Car.cgr’。 A使用‘Car’作为pool名称。该文件存储一汽车模型。
③ 从‘groups’标签中，拖动pool名称‘3D scene render’到上方的Channel视图。
③ 从‘groups’标签中，拖动pool名称‘Car’到下方的Channel视图。
③ 连接下方Channel视图中的‘Car’channel到上方Channel视图中的‘Render’channel。
③ 在‘’Animation 3D View’标签上单击。 注意一个汽车模型显示在预览窗口中。
③ 在下方的Channel视图中的‘Current Car Texture’channel上按右键并从上下文菜单中选择‘Public Channel Functions > Convert to Parameter Channel’。 注意channel变为绿色。 同时注意上方的Channel视图中蓝色的Public Call Channel‘Car : Car - > Car’现在有一个Value类型的子连接。
③ 拖动一个Value到上方的Channel视图上并重命名为‘Current Car Texture’。 将它连接到蓝色的‘Car : Car - > Car’channel。
③ 改变这个‘Current Car Texture’channel的值为‘1’。 注意在Animation 3D View窗口中汽车模型的纹理被改变了。



完成后的场景：

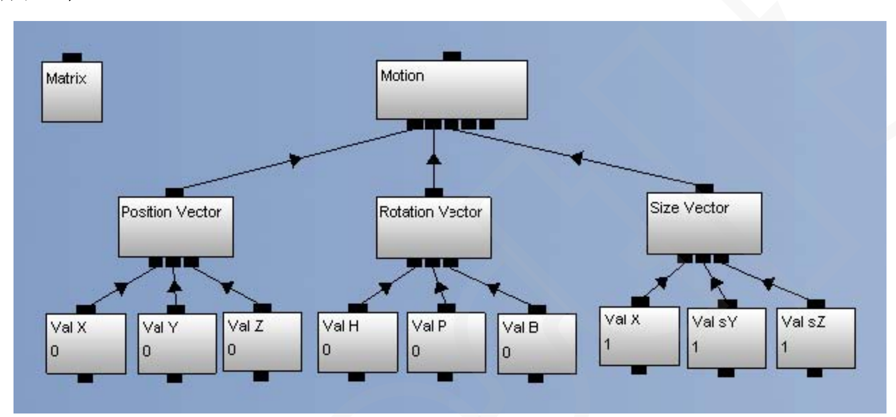
③ ..\Tutorials\3.5 – Multiple channel groups\Multiple channel groups – Complete.cgr

3.6 Mathematical operators(数学操作)

通过应用线性代数的规则，虚拟现实中的三维物体可以被移动,旋转和缩放。本章对于高级 Quest3D 用户来说，仅仅是一个介绍。要获取更多信息,请参阅描述此问题的文献。

Matrix and Motion

就像在 2.1 章讨论的，在 Quest3D 中的三维物体把它们的位置，旋转和缩放放在一个 Matrix 或 Motion channel 中。一个 Motion channel 被分成三个 Vectors 并且每个 Vector 被分成三个 Value。



Operators 运算符

在 Quest3D 中，Operator channel 允许你动态地改变这些元素中的每一个。

按一般规律,在 Quest3D 中基本 Operator channels 按照结果的 channel 类型来排列。例如，如果操作结果是一个 Value,可以在 Value Operator base channel 找到该运算符。

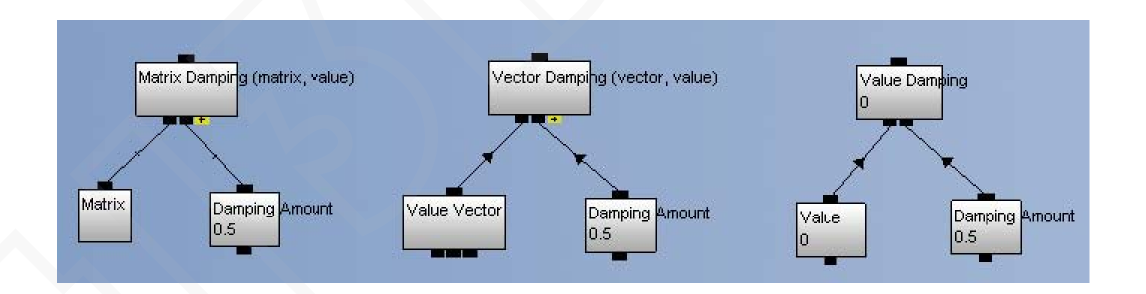
Value Operator base channel 的大多数功能用于从其他的(类型的) channel 存取数据。一个经常使用 Value Operator 的例子是‘Get distance (vector, vector)’。

Vector Operator base channel 也可以用于从其他（类型的) channels 中检索数据。一个例子是‘Get Translation (position) from Matrix’。

Vector Operators 也被用来变换矢量。变换可以包含两个矢量之间的基本加法,减法,乘法和除法。某些 Vector Operators 可用于设置一个矢量的长度，或使用一个数值或矩阵乘以一个矢量。

Matrix Operators 可用来将矢量转化为矩阵，比如‘Create Translation Matrix’。 Matrix Operators 还可以检索信息，比如‘Get Current Camera Matrix’。‘‘Matrix Interpolate’可用于融合两个矩阵。

一种特殊类型的 Operator 是‘damping’，它类似于 Values, Vectors 和 Matrices 一样存在。它根据时间缓慢改变一个变量，有效地从当前状态融合到新的状态该影响是对数性的，并且它的速度可以通过改变它的 second 子连接值来调节。该阻尼变速范围从‘0’到‘1.0’。

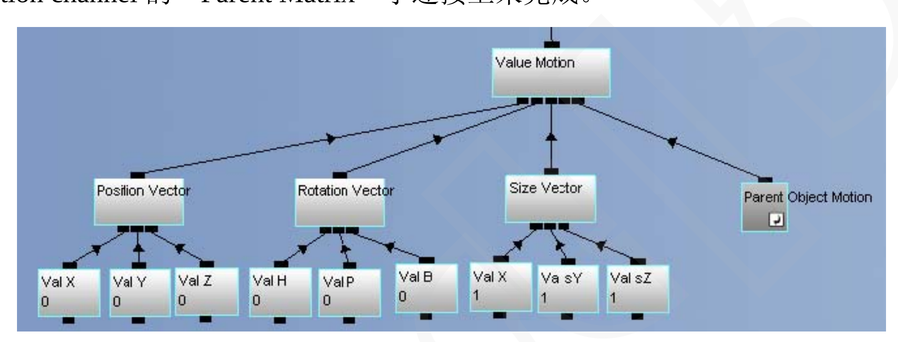


注意该‘Value Damping’Operator 是一个独立的 channel。

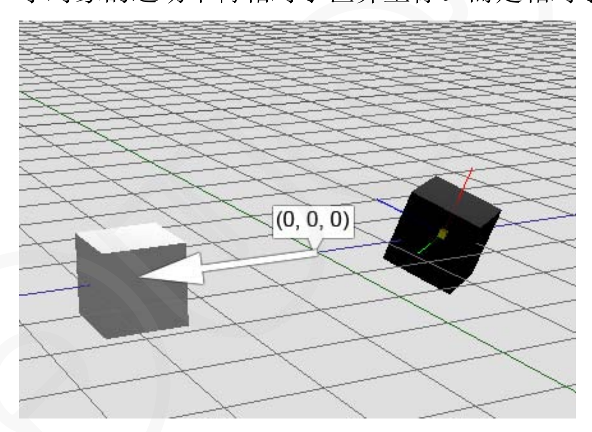
Parenting

通常，场景中三维物体的运动是相对于世界坐标原点(0, 0, 0)。然而，它也可以连接某一个对象到其它对象上。连接到第二个物体上的被称作子。连接第一个对象上的被称作父。

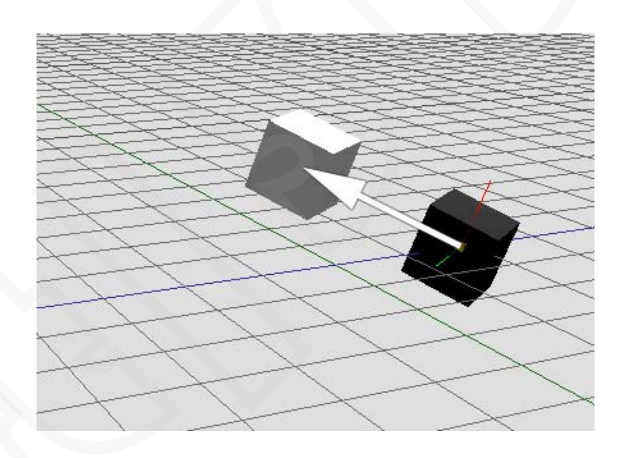
在 Quest3D 中连接一个对象到另一个，可以通过连接该父的 Motion channel 到子对象 Motion channel 的‘Parent Matrix’子连接上来完成。



子对象的运动不再相对于世界坐标。而是相对于它的父对象运动。



在上图中，亮的立方体的位置和旋转是相对于世界原点的。下图中,该亮立方被‘parented’到暗立方。该亮立方的位置和旋转相对于该暗立方。



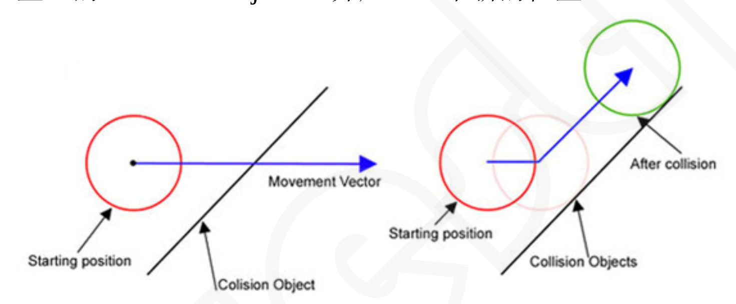
Look At

Motion channel 的第四子连接被称作‘Look At Matrix’。一个物体可以朝向另一个物体的方向，通过连接目标物体的 Motion channel 到‘Look At Matrix’来实现。沿着第一个物体的局部 Z 轴移动，该物体将朝着‘Look At’物体前后移动。

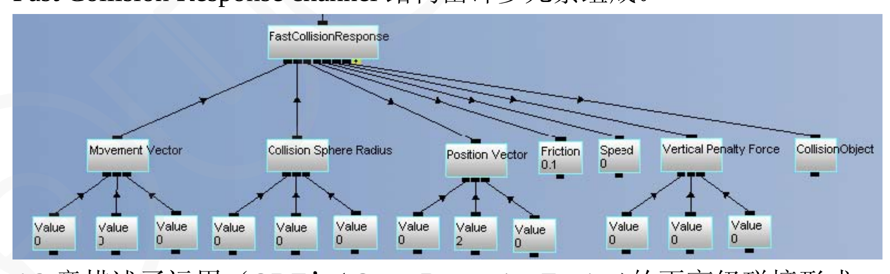
例子包含一个跟随移动汽车的像机，和一个朝着他的目标方向行走的角色。

Collision Response

Fast Collision Response channel 是一个模拟对象之间碰撞的系统。检查一个球体在它当前位置上的 Collision Objects，并产生一个新的位置。



Fast Collision Response channel 结构由许多元素组成。



4.2 章描述了运用‘ODE’(Open Dynamics Engine)的更高级碰撞形式。

实例

下面的实例为一个移动的汽车添加一个旋转车轮，演示了父与子的概念。使用一个像机

对象演示一个矩阵的 Look At 功能。

需要 quest3d 场景：

③ ..\Tutorials\3.6 – Mathematical operators\Mathematical operators 1.cgr

需要的模板：

③ Variables \ Value \ Expression Value (x2)

③ Scene \ Render Camera Light

Step by step:

③ 打开‘Mathematical operators 1.cgr’场景。 它存储一辆活动的汽车，一个自由的车轮和一个照像机模型。注意那个自由的车轮位置在(0, 0, 0)。

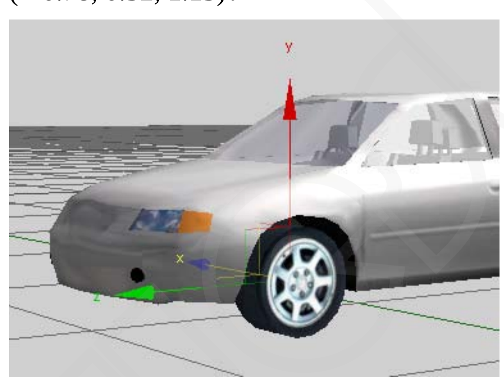
③ 创建一个‘Car Motion’channel 的快捷方式。 C 将它连接到‘Front Left Wheel Motion’的‘Parent Matrix’子连接。 注意那个车轮对象不再位于(0, 0, 0)。 它与汽车模型的位置相同。

③ 进入 Animation Section。

③ 单击 Editor Camera 按钮使用 Quest3D 编辑像机查看该场景。

③ 在列表中单击‘Front Left Wheel’对象选择它。

③ 使用 Gizmo 移动‘Front Left Wheel’调整它的位置。 或者，输入下列 x,y,z 坐标：(- 0.78, 0.32, 1.13)。



③ 进入 Channel Section。

③ 拖动一个 Expression Value 到 Channel 视图上并把它连接到‘Front Left Wheel Pitch’channel 上。

③ 双击 Expression Value 并输入下列公式:‘OLD + 0.5 * TC’。 C 单击‘OK’按钮。注意该车轮对象现在开始旋转。

③ 创建 Expression Value 的三个快捷方式并分别将它们连接到‘Front Right Wheel Pitch’,‘Rear Left Wheel Pitch’和‘Rear Right Wheel Pitch’channels。 注意所有四个车轮现在开始旋转。

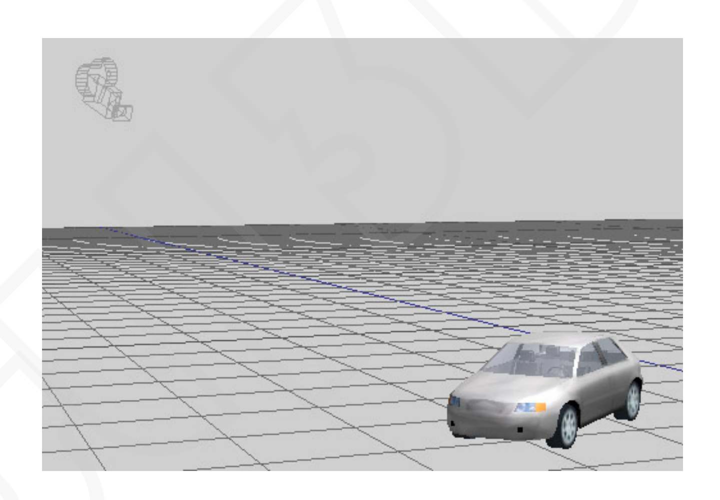
③ 创建一个‘Car Motion’的快捷方式。 将它连接到‘Camera Object Motion’的‘Look At Matrix’子连接上。 注意该三维相机模型现在朝向该汽车。

③ 进入 Animation Section。

③ 从 Timer Selection 下拉列表中，选择‘car animation’计时器。

③ 单击 Play 按钮。， 注意汽车开始沿着轨道运动。 注意相机模型一直看着该汽车。

③ 单击 Project Camera 按钮，现在场景从 Follow Camera 角度显示



完成后的场景：

- ③ ..\Tutorials\3.6 – Mathematical operators\Mathematical operators 1 – Complete.cgr

你可以保存现在的 channelgroup 到你的项目目录。如下步骤处理一个新的场景。

需要 quest3d 场景：

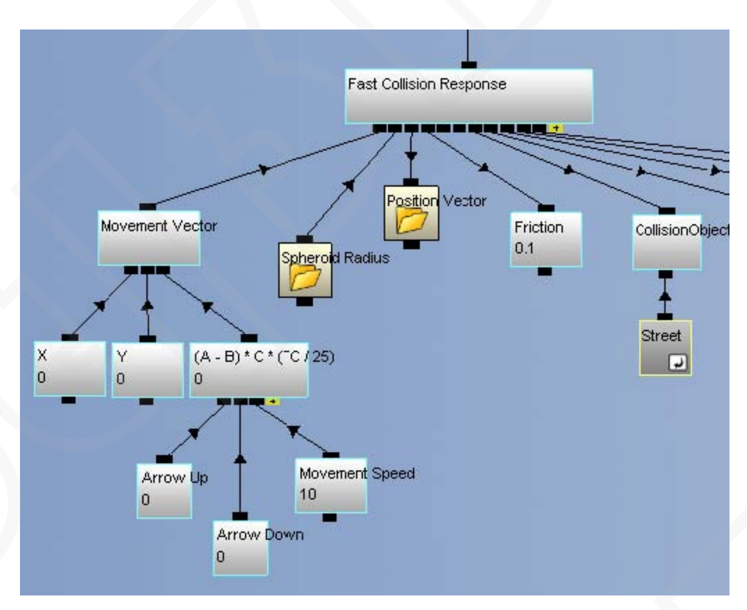
- ③ ..\Tutorials\3.6 – Mathematical operators\Mathematical operators 2.cgr

需要的模板：

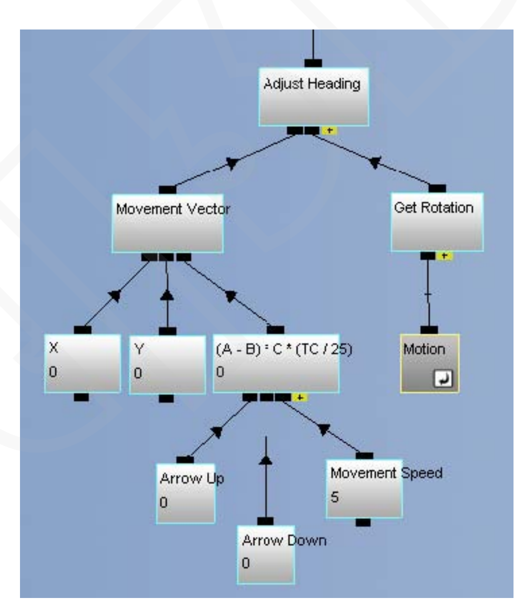
- ③ Variables \ Value \ Expression Value (x3)
- ③ Logic \ User Input \ User Input (x4)
- ③ Variables \ Value \ Value (x2)
- ③ Variables \ Vector \ Vector Operator (x2)
- ③ Variables \ Matrix \ Matrix Operator
- ③ Variables \ Vector \ Vector (using values)

Step by step:

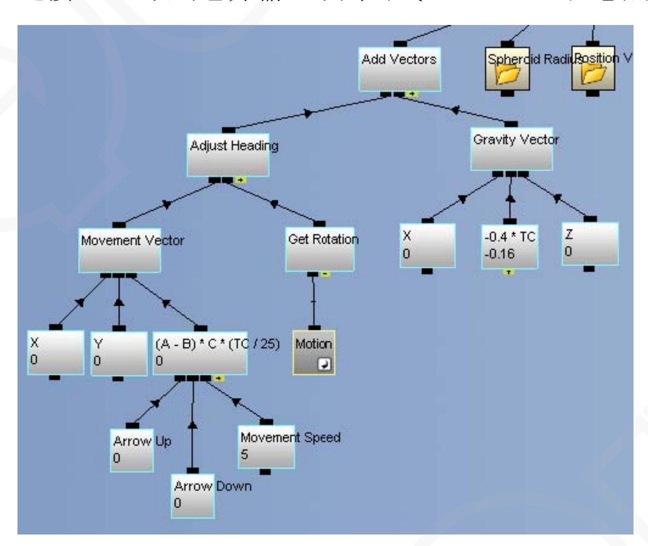
- ③ 打开‘Mathematical operators 2.cgr’场景。 它存储一个街区场景和一个弹性的球体碰撞对象。
- ③ 选择并删除‘Movement Vector’channel的‘Z’Value。
- ③ 添加一个Expression Value并连接它到‘Movement Vector’channel的第三个(现在空的)子连接上。
- ③ 拖动两个User Input channels到Channel视图上并将它们两个连接到Expression Value。分别为它们绑定向上和向下键 分别重命名它们为‘arrow up’和‘arrow down’。
- ③ 添加一个Value channel并将它连接到Expression Value。 重命名该Value为‘Movement Speed’。 双击该channel并改变它的数值为‘1’。
- ③ 双击该Expression Value并输入下列公式：(A- B) * C * (TC / 25) A是‘Arrow Up’channel, B是‘Arrow Down’channel并且C是‘Movement Speed’channel。 TTC是Tick Count，因子‘25’说明25帧/秒。



③ 切换到Run Mode。
③ 单击向上键测试该场景。注意camera视点向前移动。
③ 单击向下键向后移动。
③ 左右移动鼠标。注意视点被改变。
③ 单击向上键同时从左至右移动鼠标。 注意你同时旋转并向前移动。 同时向前移动的方向并不是照像机的指向。
③ 切换到Edit Mode。
③ 将‘Movement Vector’从‘Fast Collision Response’channel上断开。
③ 添加一个Vector Operator channel并将它连接到‘Fast Collision Response’channel。 双击它并从下拉列表中选择‘Multiply Matrix with Vector (vector, matrix)’。 关闭该窗口并重命名该channel为‘Adjust Heading’。
③ 连接‘Movement Vector’到‘Adjust Heading’channel。
③ 拖动一个Matrix Operator channel到Channel视图上。 双击它并从下拉列表中选择‘Get Rotation Matrix from Matrix (matrix)’。 关闭该窗口并重命名该channel为‘Get Rotation’。
③ 创建一个‘motion’的快捷方式并将它连接到‘Get Rotation’channel。
③ 连接‘Adjust Heading’channel到‘Motion’channel。



③ 切换到Run Mode。
③ 同时单击向上键和左键测试该场景。注意照像机视点向前移动的方向与相机的指向相同。
③ 切换到Edit Mode。
③ 改变‘Position’channel的‘Y’Value为2.注意照像机悬浮在空中。下列步骤将为该模型添加碰撞和重力。
③ 创建一个‘Environment’channel的快捷方式并将它连接到‘Collision Object’channel。
③ 将‘Adjust Heading’从‘Fast Collision Response’channel上断开。
③ 添加一Vector Operator channel。 D双击它并从下拉列表中选择‘Add Two Vectors (vector, vector)’。 C关闭该窗口并重命名该channel为‘Add Vectors’。
③ 连接‘Adjust Heading’channel到‘’Add Vectors’channel。
③ 连接‘Add Vectors’channel到‘FastCollisionResponse’channel。
③ 拖动一个Vector with Values模板到Channel视图上并把它连接到‘Add Vectors’channel。 重命名该Vector channel为‘Gravity Vector’。
③ 选择并删除‘Gravity Vector’channel的‘Y’Value。
③ 添加一个Expression Value并连接它到‘Gravity Vector’channel的第二(现在空的)子连接上。 双击它并输入下列公式：- 0.4 * TC注意照像机掉到地上。



③ 通过再次改变‘Position Vector’channel 的‘Y’Value 为 2 再测试该重力。注意该照像机再一次掉到地上。

③ 在该环境中移动来测试该场景。注意照像机视点移动的方向与相机的指向相同,并且由于重力保留在地上。

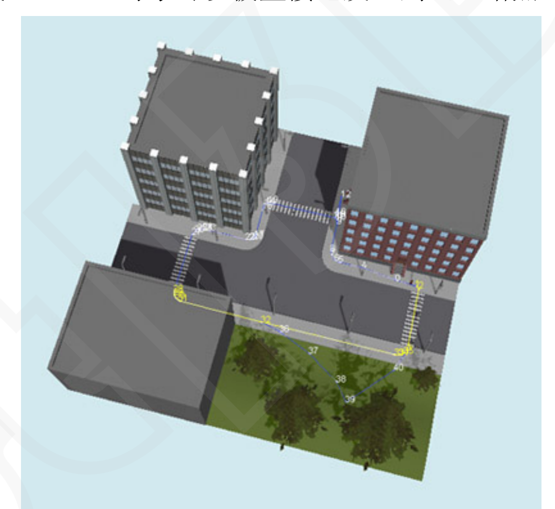
完成后的场景：

③ ..\Tutorials\3.6 – Mathematical operators\Mathematical operators 2 – Complete.cgr

3.7 Pathfinding（寻径）

通过添加行走的角色或移动的汽车，虚拟世界可以被赋予生命。这些可能是预先定义好的，但是有时需要更灵活的解决方案。角色可能不得不在红绿灯，断开的桥梁或彼此的前面停止，Pathfinding 允许动态的线路选定。

寻径依赖于在虚拟环境中设定的分段点或‘nodes’。对象可以被直接地发送到这些结点，或在路径结构之上。特别地如果环境复杂，设置一个合适的路径结构是重要的。

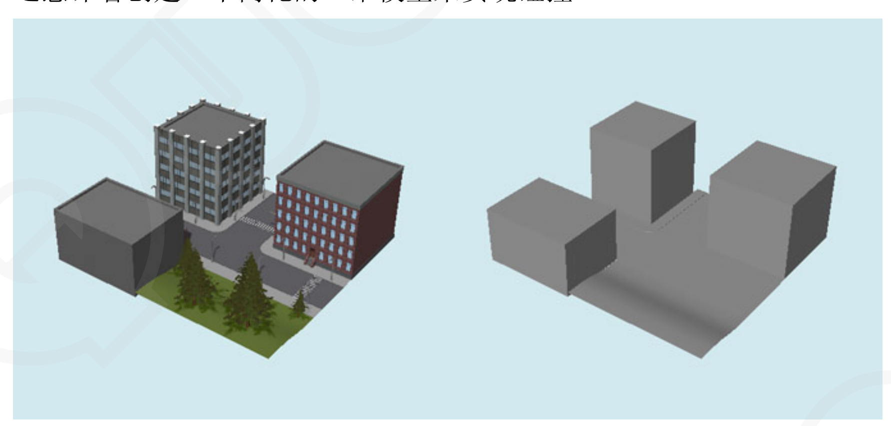


在 quest3d 中，寻径通过两个 channels 来处理。第一个是三维曲线图，它存储了所有结点和路径结构信息。另一个是 Motion Planning channel，它计算通过该路径从当前位置到目的位置的最短路径。

3D Graph

三维曲线图的结点必须被画在一个碰撞物体之上。该碰撞物体可以连接到 3D Graph channel 的第一个子连接。 T3D Graph 的另一个子连接也需要一个 Collision Object channel。该物体用于决定在路径计算期间的视线。

这些两个碰撞物体可以是相同的。建议在任何情况下，保持它们尽可能‘low poly’。 通常，这意味着创建一个简化的三维模型来实现碰撞。



Motion Planning

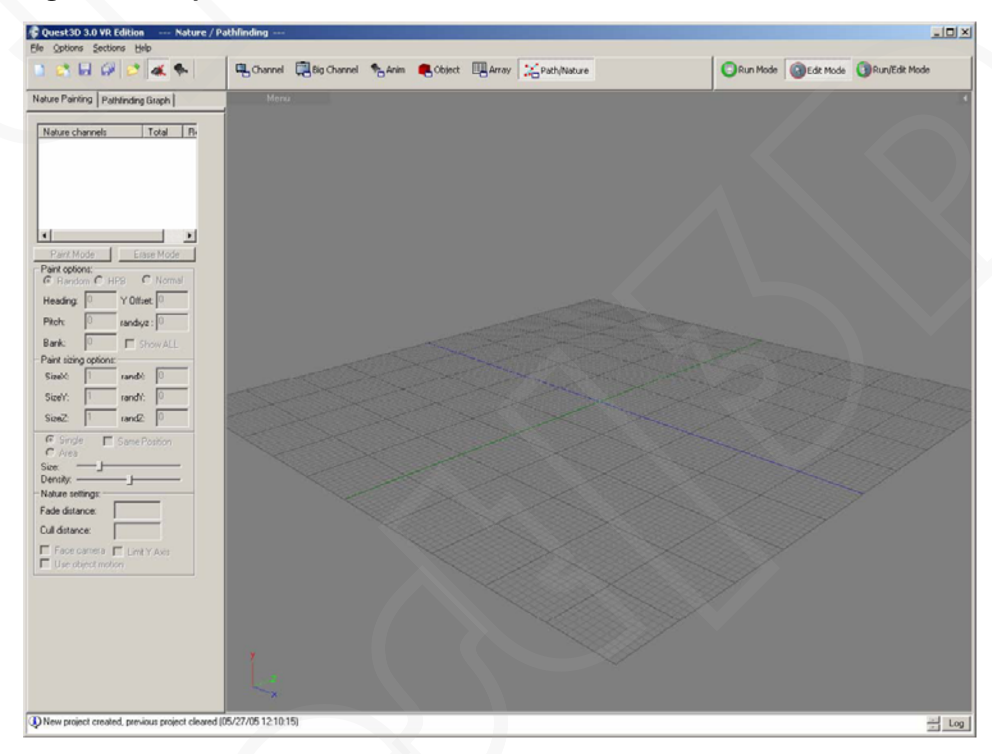
Motion Planning channel 有四个子连接

第一个子连接要求一个寻径物体的当前位置。第二个代表目的地位置。第三个是‘Next node range’。如果一个物体在任何给定结点的这个距离范围之内，它将在计算路径上寻找下一个节点。第四个是三维曲线图，它存储了所有结点和路径结构信息。

结点的位置可以使用 Get Motion Planning Info Vector channel。它要求 Motion Planning channel 作为它的第一个子连接。第二个子连接是一个 Value 它定义了在三维曲线图上哪个结点被返回。

一个寻径物体是朝着它的目标移动方向的物体。查看 3.6 章获取有关有使用 vector 移动物体的更多信息。

Pathfinding Section Layout



在 Quest3D 中结点和路径可以在 Nature / Pathfinding Section 中放置和消除。

实例

下面的实例将在一个城市场景中创建一个动画角色在若干分段点之间找到它的路径。

需要 quest3d 场景：

③ ..\Tutorials\3.7 – Pathfinding\Pathfinding.cgr

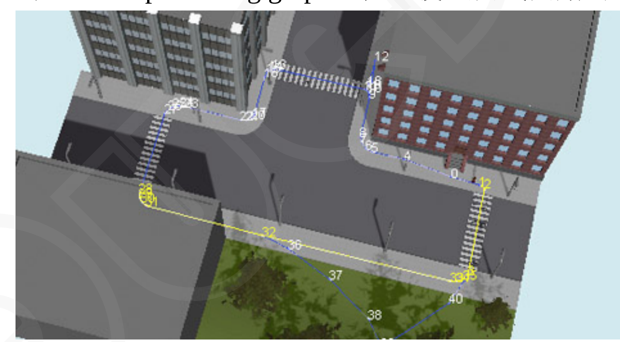
需要的模板：

③ Pathfinding \ 3D Graph
③ Collision \ Collision Object
③ Pathfinding \ Motion Planning
③ Pathfinding \ Motion Planning Info Vector
③ Variables \ Value \ Value

③ Variables \ Matrix \ Matrix Operator
③ Variables \ Value \ Array Value
③ Pathfinding \ Triggers \ Vector Proximity Trigger
③ Variables \ Value \ Set Value

Step by step:

③ 打开‘Pathfinding.cgr’文件、。 它存储一个小的城市街区场景，一个有多个动作的角色，一个跟随角色的照像机和碰撞设置。
③ 添加一个3D Graph channel并将它连接到‘Pathfinding’channel。
③ 拖动一个Collision Object到Channel视图上并把它连接到‘3D Graph’channel的第一个子连接上。
③ 创建一个‘Street’channel的快捷方式并将它连接到‘Collision Object’channel。现在准备在3D Graph上绘制结点。
③ 进入Nature / Pathfinding Section并在‘pathfinding graph’标签上单击。
③ 单击‘3D Graph’以选择它。
③ 单击‘Paint’按钮以进入结点绘制模式。
③ 在Animation 3D View上移动鼠标光标。 注意红色的圆形标记。这是结点绘制光标。
③ 在该街区环境中，找到有红色砖的建筑物，并通过左键单击人行道在门的前面放置一个结点。 得到该结点到当前绘制光标位置的一条线路。 在几米之外放置另一个结点，但仍然在该人行道上。该线路仍然连接两个结点，并且一个新的线路绘制到当前绘制光标的位置。 右键单击停止绘制。
③ 放置一些结点在该城市场景中。保持3D Graph尽可能清晰。在下图中可以看到一个恰当的曲线图的例子。
③ 当准备好的时，单击‘Idle’按钮退出结点绘制模式。注意该曲线图消失。
③ 勾选‘Keep showing graph’框以使该曲线图始终可见。



③ 进入Channel Section。
③ 添加一个Motion Planning channel。
③ 创建一个‘AI Character Position’channel的快捷方式并将它连接到‘Motion Planning’的第一个子连接上。
③ 拖动一个Motion Planning Info Vector到Channel视图上并把它连接到‘Motion Planning’channel的第二子连接上。
③ 创建一个‘Motion Planning’channel的快捷方式并将它连接到‘Motion Planning Info Vector’channel的第一个子连接上。
③ 添加一个Value并把它连接到‘Motion Planning’的第三个子连接上。 改变它的值为‘0.1’。
③ 创建一个‘3D Graph’channel的快捷方式并将它连接到‘Motion Planning’channel

的第四子连接上。

③ 拖动一个Matrix Operator到Channel视图上面并将它连接到‘AI Character Motion’ channel的‘Look At Matrix’子连接上。双击它并从下拉列表中选择‘Create Translation maxtrix (vector)’。关闭该窗口。

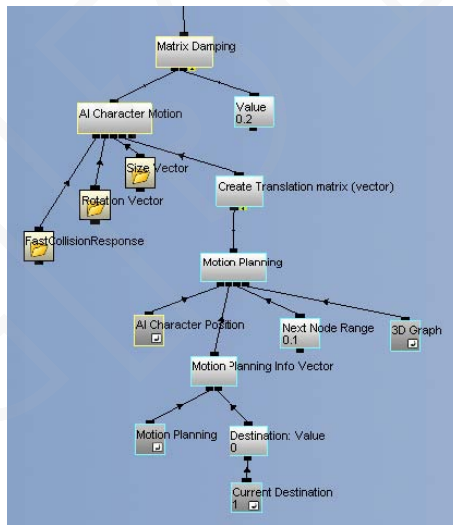
③ 连接‘Motion Planning’channel到‘Create Translation matrix (vector)’ channel 。注意该角色现在寻找节点‘0’。

③ 添加一Array Value channel并且列和channel名称使用‘Destination’。连接该channel到‘Motion Planning Info Vector’的第二子连接。

参见 3.4 章有关数组的更多信息。

③ 单击‘array manager’标签。

③ 创建其他栏从1到 5.检验你的3D Graph并插入几个结点。



③ 单击‘Channel Graph’标签。

③ 创建一个‘Current Destination’channel的快捷方式并将它连接到‘Destination’数组。

③ 进入Animation Section。

③ 切换到Run Mode。

③ 单击‘1’键测试该场景。该角色应该走向与数组字段‘1’对应结点。注意到达后，角色没有自动地停止。

③ 切换到Edit Mode。

③ 进入Channel Section。

③ 创建一个‘AI Character Position’channel的快捷方式并将它连接到‘Get Vector’channel under the‘Arrival Trigger’channel的第一个子连接上。

③ 创建一个‘Motion Planning’channel的快捷方式并将它连接到‘Get Vector’channel 的第二子连接上。

③ 创建一个‘AI Character Speed’channel的快捷方式并将它连接到‘Set Value’channel 的第二子连接上。

③ 进入Animation Section。

③ 切换到Run Mode。

③ 单击‘1’到‘5’键测试该场景。 注意该角色朝着对应结点的方向行走并在到达后停止。

③ 完成的实例存储了额外的逻辑以限制该角色的移动。打开该文件并检验它的结构。



完成后的场景：

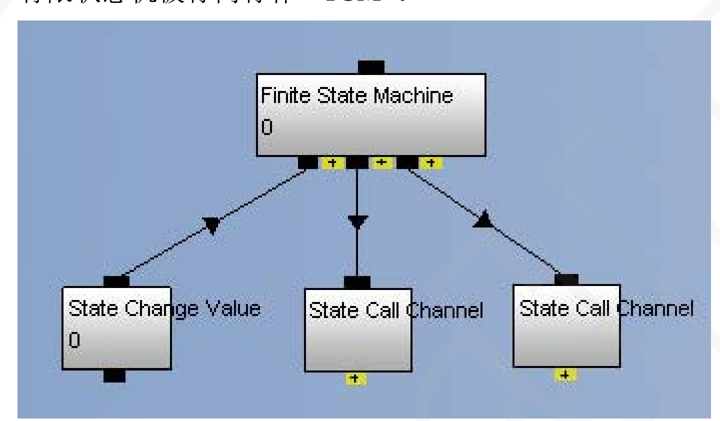
③ ..\Tutorials\3.7 – Pathfinding\Pathfinding – Complete.cgr

3.8 有限状态机

项目可能由许多子程序组成，比如各个菜单屏幕和三维场景。只要结构足够简单就可以通过使用各种各样 quest3d 逻辑 channel，在屏幕之间切换。

有限状态机 channel 可用于简化复杂结构中子程序之间切换。在上述例子中，各种各样菜单屏幕可以被认为是‘states’。

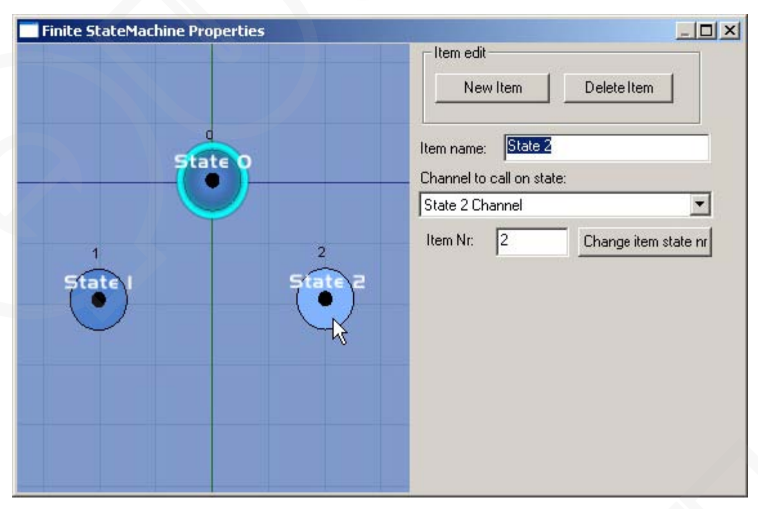
有限状态机被称简称作‘FSM’。



Finte State Machine channel 有许多子连接。连接到第一个子连接的 Value 被检查以触发状态改变。第二子连接代表实际的状态。只有当前状态被调用。最后，第三个子连接的子可以在特定状态改变时被触发

Finite State Machine Graph（有限状态机图）

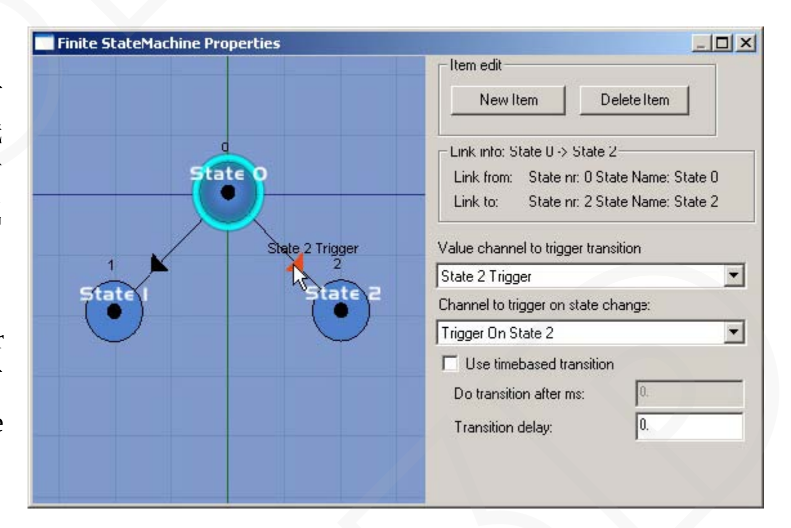
有限状态机的属性窗口看上去与 Channel 视图类似。



使用蓝色的圆形对象代表一个 FSM 状态。当前状态高亮度显示。在该屏幕的右边，显示选定状态对象的属性。

‘Channel to call on state’下拉列表存储所有连接到 FSM 的‘State Call Channels’子连

接的 channels 的名称。



连接状态对象

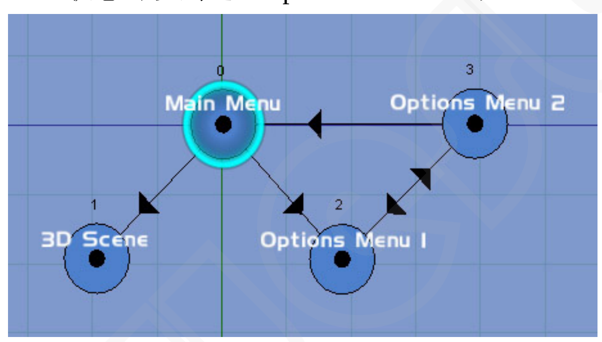
状态对象可以通过从一个黑点向另一个拖动一个线条而彼此连接。在屏幕的右边，显示选定状态连接的属性。

‘Value channel to trigger transition’下拉列表存储所有连接到 FSM 的‘State Change Value’子连接的 channels 的名称。

‘Channel to trigger on state change’下拉列表存储所有连接到 FSM 的‘State trigger channels’子连接的‘channels 的名称。

分层结构

下图中，当前状态是‘main menu’。只有‘3D Scene’和‘Options Menu 1’状态是可达的。此外，‘Options Menu 2’状态只能从‘Options Menu 1’状态可达。然而，从‘Options Menu 2’状态可以到达‘Options Menu 1’和‘Main Menu’状态。



Time based state changes

也可以创建基于时间的状态改变。该特性允许用户在一定时间(毫秒）之前移动到下一个状态上。

Examples

其他使用有限状态机的例子包含虚拟训练仿真中的阶段，高级角色动画设置和人工智能角色行为。

实例

下面的实例显示有限状态机的作用。 I 它用于一个菜单结构的导航。

需要 quest3d 场景：

③ ..\Tutorials\3.8 – Finite State Machine\Finite State Machine.cgr

需要的 channel：

③ Logic \ Finite State Machine

Step by step:

③ 打开‘Finite State Machine.cgr’文件、。 它存储一个主菜单画面和一个选项菜单画面。 同时也存储了两个非常简单三维场景，即一个红色的球体和一个蓝色的盒子。

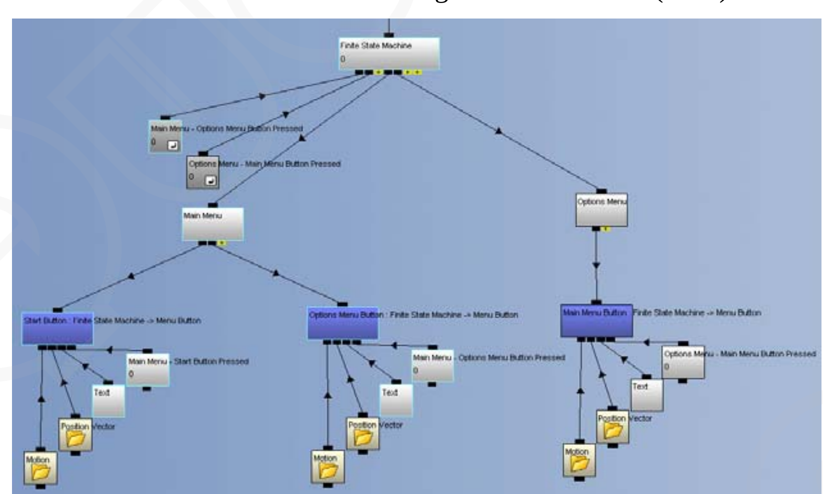
③ 拖动一个Finite State Machine channel到Channel视图上并把它连接到‘Program’channel上。 注意它的当前值或‘state’，是‘0’。

③ 连接‘Main Menu’channel到‘Finite State Machine’的‘State Call Channels’子连接。 注意‘Main Menu’channel没有被调用，因为还没有定义状态逻辑。 同时一个new‘State Call Channel’子连接被加到有限状态机，允许额外的子连接。

③ 连接‘Options Menu’channel到FSM的第二个‘State Call Channel’子连接(空的)。 注意它仍然没有被调用。

③ 连接一个‘Main Menu – Options Menu Button Pressed’channel的快捷方式到‘finite State Machine’channel的‘State Change Values’子连接。 注意一个新的‘State Change Values’子连接被加到FSM，允许额外的子连接。

③ 连接一个‘‘Options Menu – Main Menu Button Pressed’channel的快捷方式到Finite State Machine的第二个‘State Change Values’子连接(空的)。



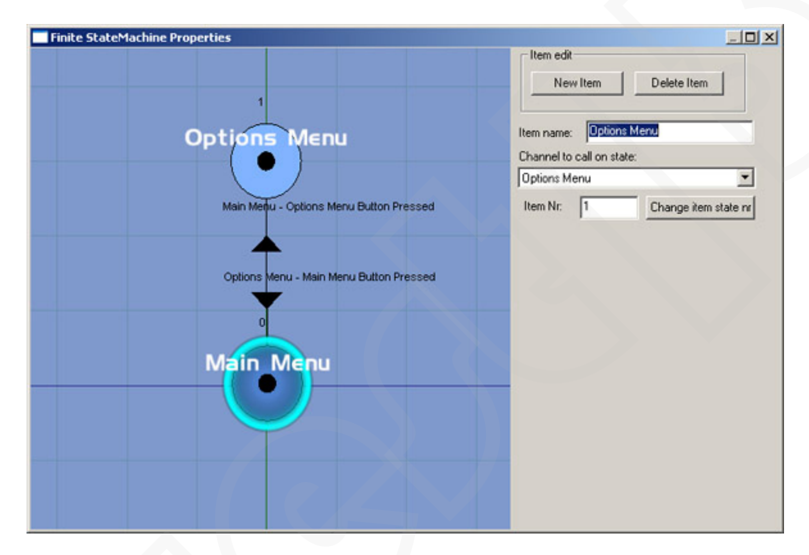
③ 双击该‘Finite State Machine’channel打开的它的属性窗口。 注意该区域类似于一个在左边的Channel视图和在右方的选项选。

③ 单击该‘new item’按钮并在FSM图中单击一次。 注意一个圆形对象被创建。

③ 单击该对象以选择它。 注意它的属性显示在该窗口的右边。 改变它的‘item name’为‘main menu’。 F从‘Channel to call on state’下拉列表中，选则‘main menu’。

注意‘Main Menu’channel现在被调用，并且该主菜单画面显示在Animation 3D View窗口中。 注意列表中的选项与连接到FSM的‘State Call Channels’子连接的channels对应。

③ 在‘main menu’对象上方创建一个新的对象并选择它。 改变它的‘item name’为‘Options menu’。 从‘Channel to call on state’下拉列表中，选则‘Options menu’。
③ 从‘Main Menu’对象中的黑点拖动一个连接到‘Options Menu’对象中间的黑点上来连接两个对象。 注意箭头的方向。
③ 单击该连接以选择它。 注意它的属性显示在该窗口的右边。 从‘Value channel to trigger transition’下拉列表中，选择‘Main Menu－ Options Button Pressed’。 注意现在Channel视图中同名的Value被调用。 注意列表中的选项与连接到FSM的‘State Change Values’子连接的channels对应。
③ 从‘Options Menu’对象拖动一个连接到‘Main Menu’对象。 选择该连接并从‘Value channel to trigger transition’下拉列表中，选择‘Options Menu－ Main Menu Button Pressed’。



③ 关闭该有限状态机属性窗口。
③ 进入Animation Section。
③ 切换到Run Mode。
③ 单击Animation 3D View窗口中的‘Options’按钮。 注意选项菜单屏幕被显示。
③ 单击‘Main Menu’按钮回到主菜单。
③ 再次单击‘Options’按钮调出选项菜单。
③ 切换到Edit Mode。
③ 进入Channel Section。
③ 双击该Finite State Machine channel打开的它的属性窗口。 注意当前状态，‘options menu’，现在高亮显示。
③ 关闭FSM窗口。 I在Channel视图中，注意该有限状态机的值现在是‘1’。 注意该‘options menu’channel被调用，而‘main menu’channel没有。 注意‘Options Menu － Main Menu Button Pressed’Value高亮显示，而‘Main Menu－ Options Button Pressed’Value没有。
③ 连接‘Scene Selection Menu’channel到FSM的第三个‘State Call Channel’子连接(空的)。
③ 连接‘Main Menu－Start Button Pressed’channel到FSM channel的第三个‘State

Change Values’子连接(空的)。

③ 连接‘Scene Selection Menu－ Main Menu Button Pressed’channel到FSM channel的第四‘State Change Values’子连接(空的)。

③ 打开的FSM属性窗口。

③ 添加一新的对象，‘Scene Selection Menu’。 设置它调用‘Scene Selection Menu’channel。

③ 连接‘Main Menu’对象到‘Scene Selection Menu’对象并正确的数值改变设置它。

③ 连接‘Scene Selection Menu’对象回到‘Main Menu’对象并以正确的数值改变设置它。

③ 关闭FSM窗口。

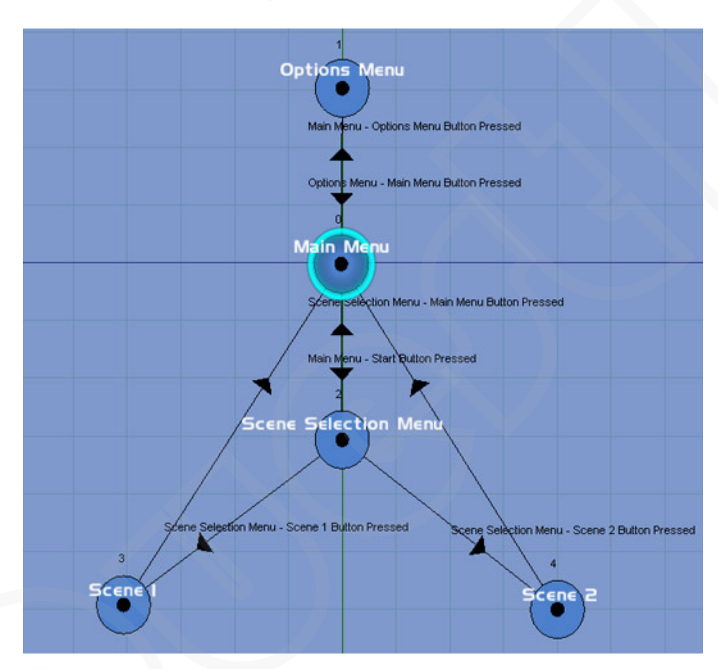
③ 进入Animation Section。

③ 切换到Run Mode。

③ 通过单击到目前为止创建的有作用的按钮测试场景。

③ 切换到编辑方式。

③ 为这两个三维场景创建有限状态机逻辑。 所有需要的channels都已经存在的Channel视图上，只须正确地连接它们。



完成后的场景：

③ ..\Tutorials\3.8 – Finite State Machine\Finite State Machine – Complete.cgr

第四部分 高级

摘要

第一部分介绍了 Quest3D 的基本概念 第二和第三部分分别详细描述了三维物体和动画，Quest3D 编程。 该部分介绍了一些高级特性。

4.1 项目管理

本章概括的讨论了 Quest3D 的开发过程。

4.2 物理模拟

ODE 是 Open Dynamics Engine（开放动态引擎）它用于物理仿真。

4.3 连接数据库

大量的信息可以被存储在数据库中。 Quest3D 支持多种类型的连接，需要 VR 版或企业版才能使用该功能。

4.4 联网

相互连接多个计算机到被称为’联网’。本章详细介绍了该主题。需要 VR 版或企业版才能使用该功能。

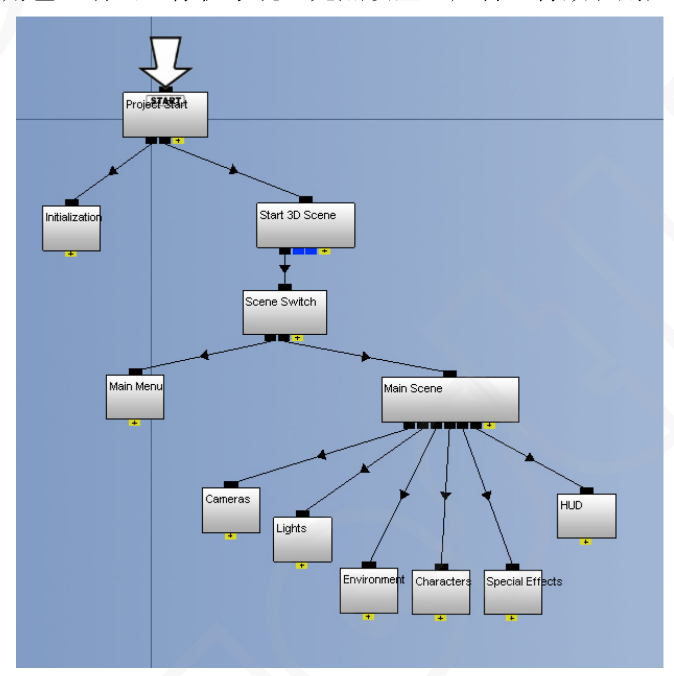
4.5 Lua 脚本

Lua 扩展了标准的 *Quest3D channel* 集。

4.1 项目管理

高级的 Quest3D 程序化可能是复杂的工程。 本章讨论项目管理。特别描述了使用好的文件结构的好处。

复杂的 Quest3D 程序是由多个不同的子程序组成的。典型的子程序可能包含：3D 环境，3D 角色，动画，像机系统，光照设置，声音，特效和用户接口。



预先创建一个工程结构有助于项目规划。 可以将一些空的 channel 组连在一起组成应用程序的主干。 应用程序总的来说更好读——对于所有的团队成员来说它更易懂。

这个结构不仅能够帮助我们预测将需要使用哪些元素，同时还使得开发计划的制定变得更加容易。可以给整个工程的子任务分配特定的成员和明确的工期。

建模人员可以将物体导入到 Quest3D 中并调整它的外观。 音效师可以开发和测试音乐和声音计划。一些专业程序员可以编写不同的子程序而不会相互干扰。

在开发过程中，任何一个子程序都会被为其工作的开发人员提供的新版所更新。

未来的工程能够从极大地从公司标准的程序结构中获利。在大多数情况下，大量的子程序可被重用。 3D 对象能够非常容易的使用新的内容来替换。元素如用户接口可能在类型和使用上是一致的。

实例

下面的实例提供项目管理的协助。 这些步骤并不是固定的，因为每一个工程都有它自己特有的问题。 然而，一般来说，通读下面的要点将是一个很好的练习。

Step by step:

- 在一张纸上写下工程的名称。
- 用一句话，描述工程的要点。 是否为一个训练程序？ 产品展示？ 建筑预演？ 一个游戏？ 还是其他类型的应用？
- 根据预想的搭建一个 3D 场景的基本框架。
- 制作一个清晰的图形用户接口框架。
- 编辑所有需要使用的元素的列表。 首先集中处理基本的子程序例如环境，像机系统和用户接口。 然后，为每个部分制作一个详细的列表，包括特定的功能，动画，材质贴图等。
- 在 Quest3D 中创建一个基本的框架结构。 基于各个子程序创建 channel 组并将它们连接在一起。 使用清晰和一致的名称。
  在创建基本框架结构时写下你能想到的所有扩展元素。
- 初步估计创建所有的元素需要多长时间。
- 写下参与该工程的所有团队成员的名称。
- 基于各个成员的职责将子程序和 3D 对象分组。添加成员并写下每个成员需要的总时间。
  考虑自程序间的依赖关系。 某些部分必须在其他部分完成后才能开始。 整理列表以便提出一个可行的计划。
- 与团队的有关成员讨论该计划。
- 如果程序过大，将计划分割为三个或四个部分。 每个部分可以用于和客户签订合约以便监督进度。
- 确定计划。
- 如果有必要，写一个小的工程提议讨论该应用程序。 如果有的话包含一个或两个适当的截图。
- 与客户讨论该计划和工程提议，调整计划。
- 开始开发。 经可能严格按照计划。
- 注意添加新的特性将增加生产指标的时间，并削减应用程序必要的设计。
  尤其是添加客户提出的特性。 与相关的成员讨论并听取他们的意见。 如果需要，向客户建议可选的方案。 并告知客户添加特性将影响开发计划，并且很可能需要额外的费用。
- 如果可能，设计一个用户测试。 邀请一些与项目无关的人来测试程序。 记录问题和特别好的方面。 如果需要，改变设计。
  在程序完成或快要完成时，给客户展示并写下最后需要改变的地方。
- 用一到两天来细化程序。
  发布 1.0 版。
- 项目结束后，与整个团队讨论开发过程。
- 预料客户的反馈并准备提供支持。

③ 记录开发过程中好的方面和错误的地方。

③ 记录所有的 3D 对象和子程序，可能在以后的项目会用到它们。

③ 如果在两个项目中间有时间，让团队成员优化他们的代码和 3D 模型以便用在未来的工程中。 子程序可以被转化为 Quest3D 模版。

③ 如果时间允许，让团队成员阅读新的 channel 原理和功能并测试以提高他们的技巧。准备下一个工程。

4.2 物理仿真

传统动画的一个缺点是：它们是预先定义的。 基于时间的动画可以重放，但是不会对环境的改变做出反应。

作为预定义动画的替代，实时物理引擎可以模拟真实世界的运动和行为。 物体的位置和旋转可以基于一些属性例如形状，质量和摩擦不停的计算。 外力如重力和相互之间的碰撞也会影响物体的运动。

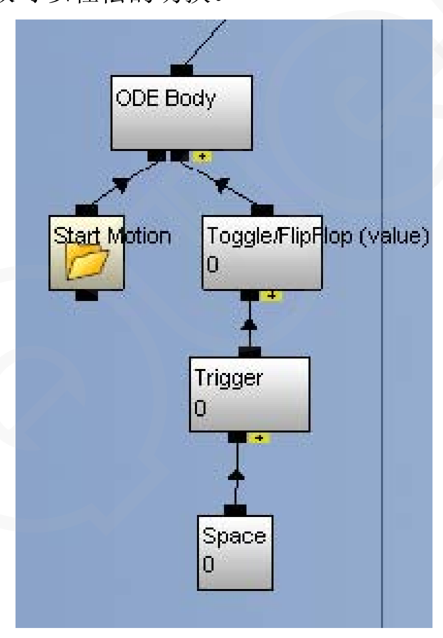
ODE

Quest3D 支持**'Open Dynamics Engine' (ODE)（开放式动力学引擎）**在 Quest3D 中可以相当容易的设置物理模拟，并且会产生非常好的效果。

Channels

ODE Body channel 是任何使用动力学项目的核心。它代替了常规的物体 *Motion channel* 结构。

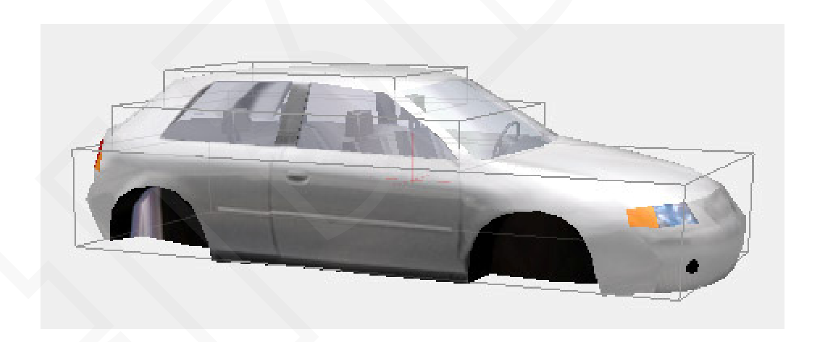
ODE Body channel 的第一个子连接是开始动作前的间隙。 连接到第二个连接上的 0 和 1 值分别禁用和启用模拟。实时动力学必须在每个物体上启用或关闭。使用 *Toggle/FlipFlop* 模版可以轻松的切换。



ODE Shapes

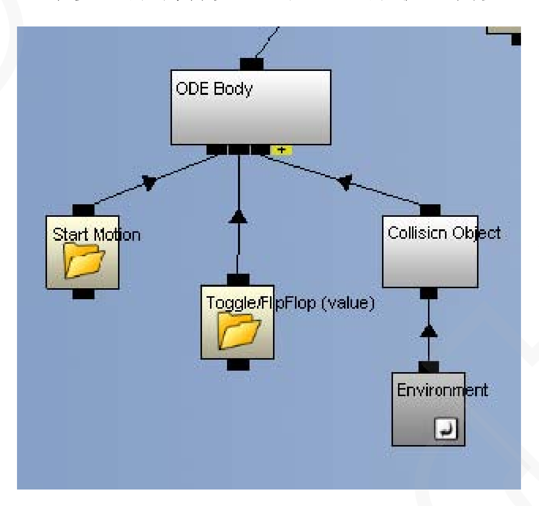
可以赋予 ODE 物体四个模拟形状中的一个。长方体，球体，平面和多边形物体。其中，只用长方体和球体使用真正动力学。平面和多边形只用作静态碰撞物体，例如环境。

复杂的动态形状可以通过添加多个长方体和（或）球体来近似。



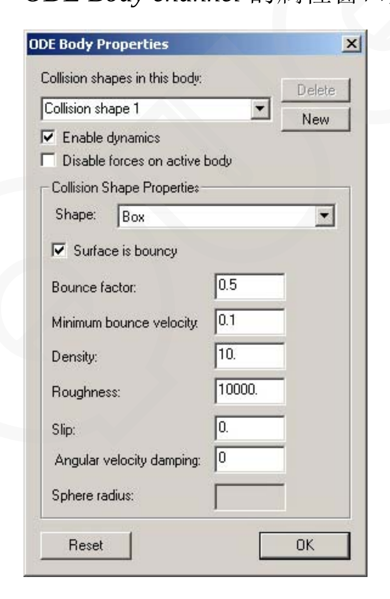
多边形

一个多边形物体类型的 ODE 形状，需要一个碰撞物体连接到它的第三个子连接上。



ODE Properties

ODE Body channel 的属性窗口列出了大量的属性。



ODE Body channel 的所有属性都可以在 Animation Section 的'Dynamics'标签中设置。只有关闭模拟时 ODE 的属性才能被编辑。

各种属性的完整描述可以在参考手册中找到（在 Quest3D 中单击 F1）。 Quest3D 中三个 ODE 物体模版代表了常用的属性集。 *Stable ODE Body*, *Slippery ODE Body* 和 *ODE*

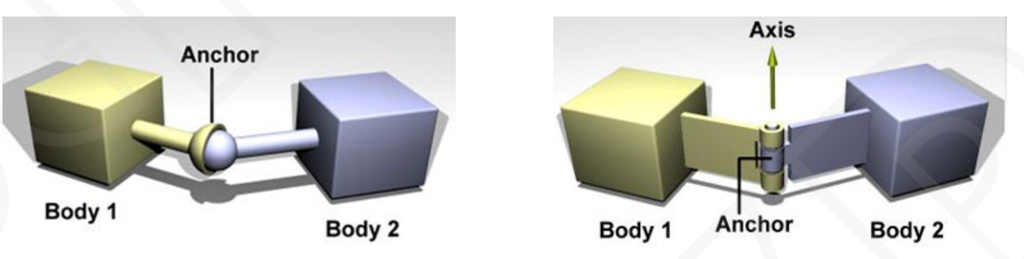
Landscape

ODE 连接

开放式动力学引擎的一个强大特性是它的连接功能。 *ODE Joint channel* 可以用来连接多个 ODE 形状。 只有长方体和球体可以作为连接的部分，平面和多边形物体不行。

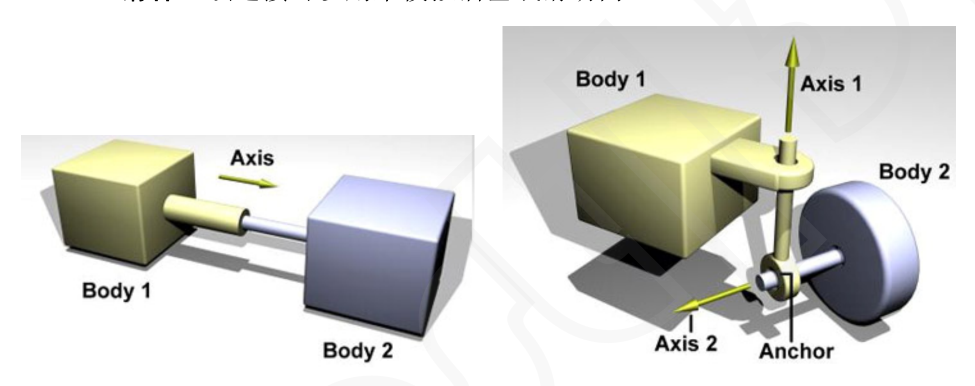
Quest3D 支持四种类型的连接。

Ball and socket（球和球窝）该连接可以用来模拟天花板上悬挂的灯。



Hinge（铰链）该连接可以用来模拟门的铰链。

Slider（滑杆）该连接可以用来模拟活塞或滑动门。



Hinge2 该连接可以用来模拟车轮的悬挂，包括弹簧。 Axis 1 用于方向，Axis 2 用来移动。

ODE 连接规则

③ 使用连接时遵守如下规则：

③ 连接的两个物体不能相互碰撞。

③ 两个物体只能使用一个连接。两个相同物体间的多个连接将出现矛盾，并导致不稳定的模拟。

③ 使用特定的连接类型可以限制自由移动。 例如滑动连接不能沿着它的轴旋转。

ODE 命令

ODE Command channel 可以用来改变模拟设置，包括精度，速度，重力和摩擦力。*ODE Command channel* 也可以用来在物体上添加动态力。 这些力可以是绝对的（例如风），或者是相对于物体的（如火箭推进器）。

ODE 信息

ODE Info Value 可以用来获取信息，如力，速度和连接旋转角度。

实例

练习的目的是熟悉在 Quest3D 中使用物理模拟。 在第一个场景中，创建了一个 ODE 行走像机。第二个实例的结果是一辆可控的汽车。

需要的 Quest3D 场景：

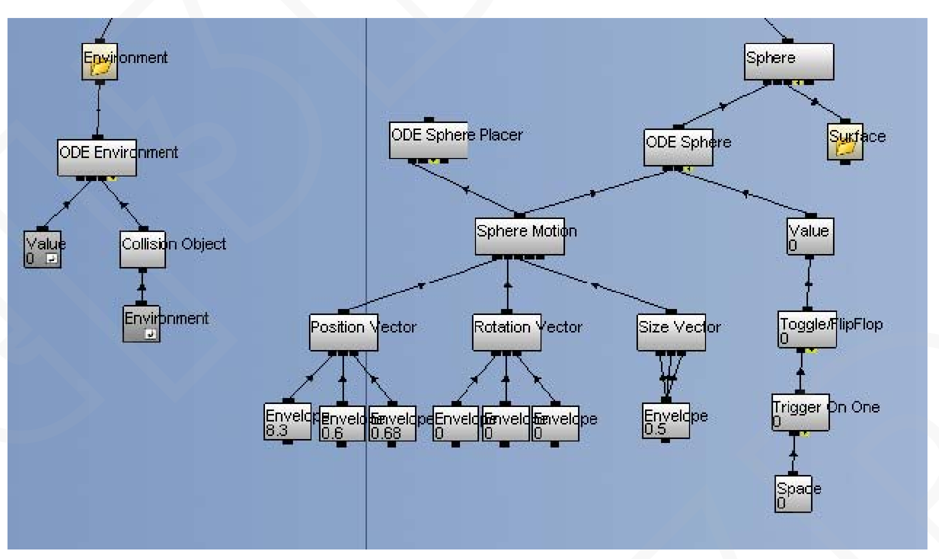
③ ..\Tutorials\4.2 – Physics simulation\Physics simulation 1.cgr

需要的模版：

③ Physics \ ODE Body
③ Objects \ Collision \ Collision Object
③ Logic \ Triggers \ Toggle Value
③ Objects \ Null Object
③ Physics \ ODE Command

Step by step:

③ 打开文件，‘Physics simulation 1.cgr’。 包含一个环境，大量的长方体，一个球体和一个像机。
③ 将‘Sphere Motion’ channel 从‘Sphere’ channel 上断开。
③ 拖动一个 *ODE Body channel* 到 Channel 视图中并将它连接到‘Sphere’ channel 的第一个子连接上。 重命名 ODE Body 为‘ODE Sphere’。
③ 双击‘ODE Sphere’ channel 打开属性窗口。 从‘Shape’下拉列表中选择‘Sphere’选项。单击‘OK’。
③ 在 ODESphere 属性窗口中，设置‘Roughness’为 50，‘Angular Velocity Damping’为 0.1，‘Sphere Radius’为 0.5。 单击‘OK’。
③ 双击‘ODE Environment’ channel 打开属性窗口。 从‘Shape’下拉列表中选择‘Polygon Object’选项。 单击‘OK’。
③ 拖动一个 *Collision Object channe*l 到 Channel 视图中并将它连接到‘ODE Environment’ channel 的‘Collision Object’子连接上。
③ 创建 ‘Environment’ channel 的快捷方式并将它连接到 Collision Object。
③ 双击‘Collision Object’ channel 打开属性窗口。 单击‘Create Tree’按钮初始化该 channel，并单击‘OK’。
③ 将'Toggle/FlipFlop'连接到’ODE Sphere’ channel 的‘Enable Body’子连接上。
③ 创建一个‘Toggle/FlipFlop’ channel 的快捷方式并将它连接到‘ODE Environment’ channel 的‘Enable Body’子连接上。
③ 将‘Sphere Motion’连接到’ODE Sphere’ channel 的‘Start Motion’子连接上。
③ 拖动一个 *Null Object* 到 Channel 视图并重命名为‘ODE Sphere Placer’。
③ 选择并删除连接到‘ODE Sphere Placer’ channel 上的‘Motion’文件夹。
将 ‘Sphere Motion’ channel 连接到 ‘ODE Sphere Placer’ channel 。



③ 进入 Animation Section。

③ 切换到 Run / Edit Mode。

③ 在屏幕左侧的列表中单击‘ODE Sphere Placer’ 物体以选择它。 移动该物体到地面上方一点。 注意现在‘Sphere’物体和它一起在移动。

③ 单击空格测试场景。 注意球体掉在地上。

③ 再次单击空格停止物体模拟。 注意球体被重置未开始位置。

③ 移动‘ODE Sphere Placer’到地面以上一点，使其恰好在地面上。
切换到 Edit Mode。

③ 进入 Channels Section。

③ 创建一个‘ODE Sphere’ channel 的快捷方式并将它连接到‘Get Movement from Matrix’ channel。 该操作使得 Sphere 的 position vector 成为球体的子。

③ 添加一个 ODE Command channel 并将它连接到‘Forces’ channel。 双击打开属性窗口。 从下拉列表中选择‘Add Force (ODE Body, vector)’选项。 选择‘Update channel name’选项单击‘OK’。

③ 创建一个‘ODE Sphere’ channel 的快捷方式并将它连接到‘Add Force (ODE Body, vector)’channel 的第一个子连接上。
将‘Force Vector’ channel 连接到‘Add Force (ODE Body, vector)’ channel 的第二个子连接上。 注意‘Force Vector’由键输入 channel 组成并由像机导向修正。

③ 进入 Animation Section。

③ 在 Animation 3D View 窗口中使用投影像机察看场景。

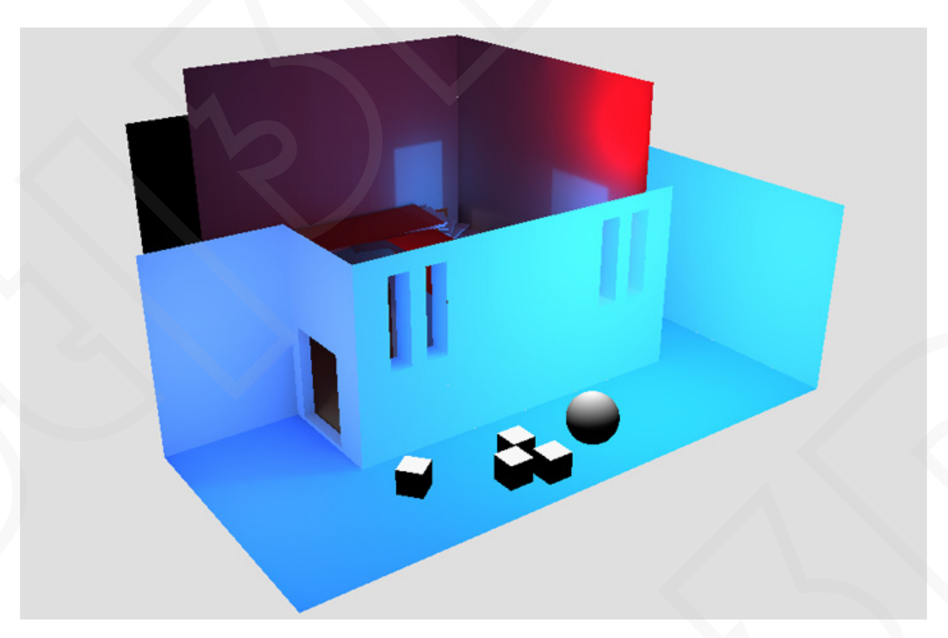
③ 切换到 Run Mode。

③ 单击空格键激活模拟。

③ 使用上下键和鼠标移动。 向斜坡上移动注意会滑下来。 向长方体移动，它们会被推开。

③ 单击空格两次停止并重新开始模拟。

③ 切换到 Edit Mode。



完成后的场景：

- ③ ..\Tutorials\ 4.2 – Physics simulation\Physics simulation 1 – Complete.cgr

保存当前工程。 下面的步骤将处理一个新的场景。

需要的 Quest3D 场景：

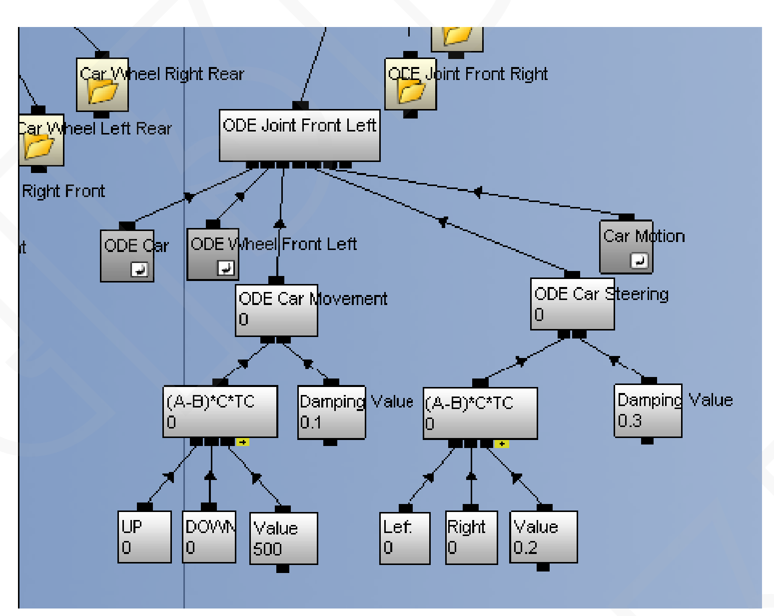
- ③ ..\Tutorials\4.2 – Physics simulation\Physics simulation 2.cgr

需要的模版：

- ③ Physics \ ODE Joint

Step by step:

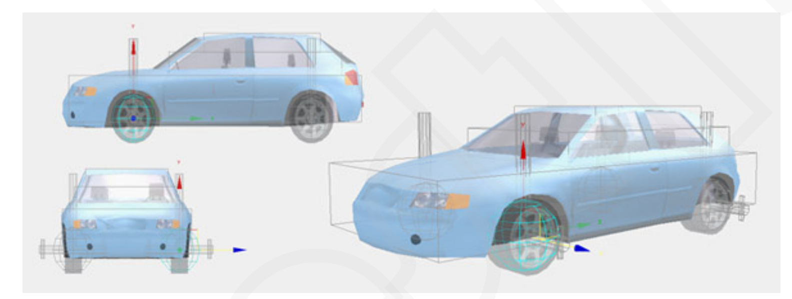
- ③ 打开文件，'Physics simulation 2.cgr'。 它包含一个环境，一个汽车和四个分开的车轮。 每一个物体已经被设置为一个 ODE 体。
- ③ 拖动一个 *ODE Joint channel* 到 Channel 视图并连接到'ODE Joints' channel。 重命名该 *ODE Joint* 为'ODE Joint Front Left'。 双击打开属性窗口。 选择'Hinge2'做为连接类型。 选择'Enable motor'选项单击'OK'。
- ③ 创建一个'ODE Car' channel 的快捷方式并将它连接到'ODE Joint Front Left' channel 的'ODE Body 1'子连接上。
- ③ 创建一个'ODE Wheel Front Left' channel 的快捷方式并将它连接到'ODE Joint Front Left' channel 的'ODE Body 2'子连接上。
- ③ 将'ODE Car Movement' channel 连接到'ODE Joint Front Left' channel 的'Velocity'子连接上。
- ③ 将 'ODE Car Steering' channel 连接到 'ODE Joint Front Left' channel 的 'Velocity2/Rotation (Hinge2)'子连接上。
  创建一个'Car Motion' channel 的快捷方式并将它连接到'ODE Joint Front Left' channel 的'Parent Matrix'子连接上。 现在的 channel 结构看起来应该像下面一样。



③ 进入 Animation Section。

③ 单击‘ODE’标签。

选择‘ODE Joint Front Left’。 移动连接的中心到左前轮的中心上。 设置作者：微软用户看起来应该像下面一样。 注意其他三个轮子已经设置了适当的位置。



③ 切换到 Run / Edit Mode。

③ 单击空格键开始物理模拟。 注意汽车掉在地上。

③ 使用方向键驾驶汽车。 注意轮子的悬挂系统表现的非常好。

切换到 Edit Mode。



完成后的场景：

③ ..\Tutorials\ 4.2 – Physics simulation\Physics simulation 2 – Complete.cgr

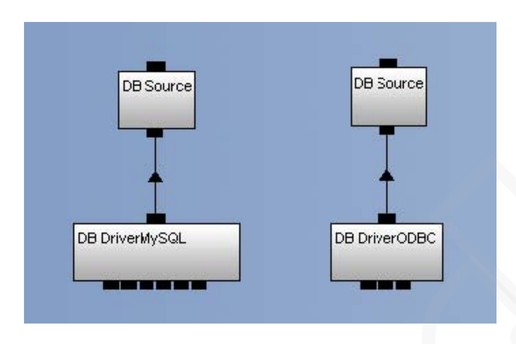
4.3 数据库连接

Qeust3D 提供两种类型的数据库连接。 第一个为默认的 MySQL，第二个为 ODBC 3.0。这两个连接类型覆盖了大部分现存的数据库技术。

注意只有在 Quest3D 企业版和 VR 版中数据库连接才是可用的。

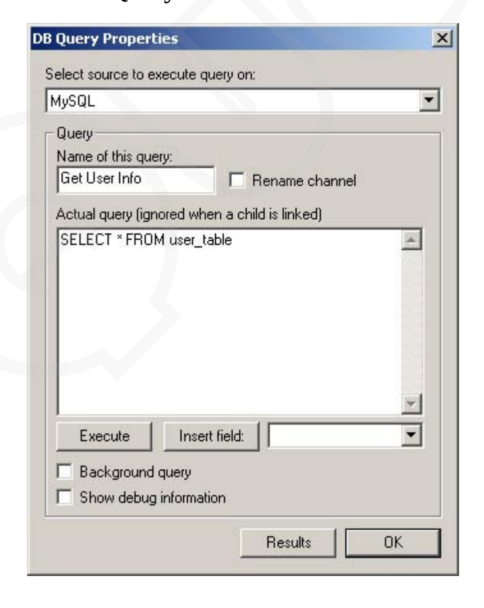
Quest3D 中数据库功能的核心在于到外部数据库的连接。 DB Driver MySQL 和 DB Driver ODBC channel 分别使用各自的协议来设置一个数据库连接。

DB Source channel 代表它自己的实际连接。



查询

建立了 MySQL 连接后，可以使用 DB Query channel 来存取数据。 在它的属性窗口中，必须设置数据源和查询的名称。 可以在对话框中输入实际的查询，或者作为 Text channel 连接到 DB Query channel 的第一个子连接上。



参数查询

DB Query channel 的第二个子连接，‘Input’可以用来向查询中添加参数。 第一个‘Input’

子作为‘$$1$$’，第二个为‘$$2$$’，等等。

常用的 MySQL 命令例子如下。

③ SELECT [column] FROM [table] WHERE [condition]

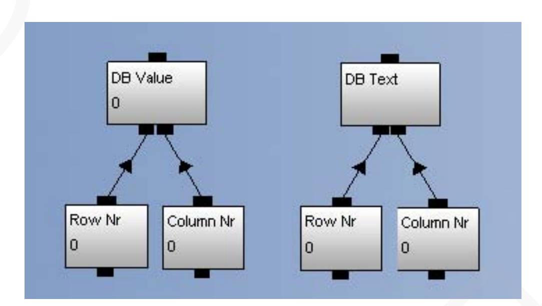
③ INSERT INTO [table] ([column1, column2]) VALUES [“value1”, “value2”])

③ UPDATE [table] SET [column=”value”] WHERE [condition]

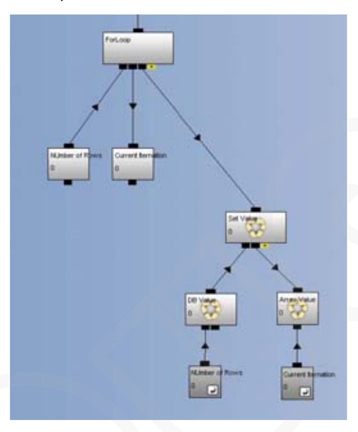
查阅 MySQL 的官方文档获取查询命令和表达式的更多信息。

在 Quest3D 中存储数据

查询的结果可以存储在 DB Value 和 DB Text 类型的 channel 中。这两种 channel 类型的属性窗口允许你选择查询和字段。 这两个 channel 的子连接可以用来指定行列数。



使用 For Loop 结构可以取回数据集，例如将它作为不同的行存贮在适当类型的 Array channel 中。



ODBC

在 ODBC 的使用方法与 MySQL 连接相同，除了连接通过驱动而不是直接连接到数据库。ODBC 驱动可以设置为连接到 dBase, MS Excel 和 MQIS 文件，也可以连接到 MS Access 数据库。

查询 ODBC 的官方文档获取查询命令和表达式的更多信息。

实例

下面的步骤使用 Quest3D channel 访问远程 MySQL 数据库。第二个例子添加数组功能。

注意： 本实例中需要访问本地或远程 MySQL 服务器。 下面的步骤依靠的数据库名为‘quest3d’，包含一个名为‘player’的表，该表中含有列‘id’，‘name’和 ‘type’ 为了取得好的结果，该表应该包含三个或更多的记录。 可以调整这些记录为你的个人信息。

需要的 Quest3D 场景：

③ ..\Tutorials\4.3 – Database connectivity\Datase connectivity 1.cgr

需要的模版：

③ Enterprise \ Database \ DB Driver MySQL
③ Variables \ Text \ Text (x3)
③ Enterprise \ Database \ DB Source
③ Enterprise \ Database \ DB Info Value
③ Enterprise \ Database \ DB Query
③ Enterprise \ Database \ DB Text
③ Variables \ Value \ Value

Step by step:

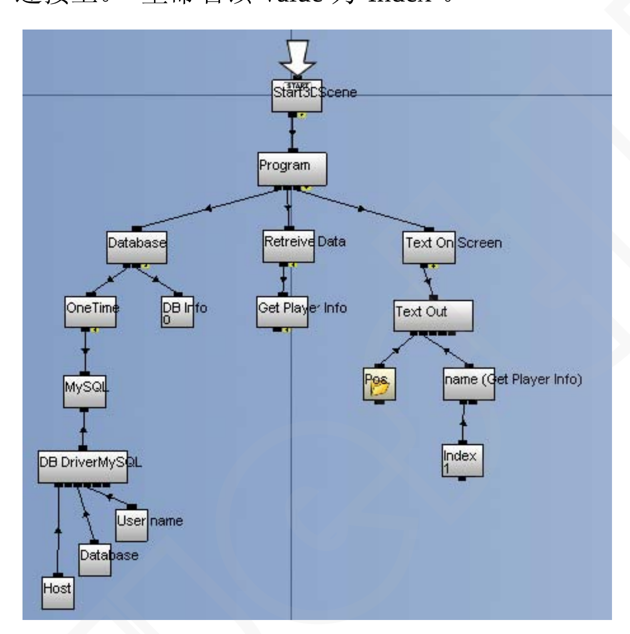
③ 打开文件，‘Database connectivity 1.cgr’。 它包含一个简单的逻辑结构。
③ 拖动一个 DB Driver MySQL channel 到 Channel 视图中。
③ 添加一个 Text channel 并将它连接到‘DB Driver MySQL’ channel 的‘Host’子连接上。重命名该 Text channel 为‘Host’。 双击该 channel 并改变文本为‘127.0.0.1’。 这是一个‘localhost’IP 地址。 如果你要访问远程数据库，使用远程服务器的 IP 地址。 单击‘OK’接受并关闭属性对话框。
③ 拖动一个另一个 Text channel 到 Channel 视图中并将它连接到‘DB DriverMySQL’ channel 的‘Database’子连接上。重命名该 Text channel 为‘Database’。双击该 channel 并改变文本为‘example’。这是远程 Quest3Dweb 服务器的数据库的名称。单击‘OK’接受并关闭属性对话框。
③ 添加另一个 Text channel 并将它连接到‘DB Driver MySQL’ channel 的‘Username’子连接上。重命名该 Text channel 为‘User name’。双击该 channel 并改变文本为‘root’。单击‘OK’接受并关闭属性对话框。
③ 如果服务器需要密码，添加另一个 Text channel 并将它连接到‘Password’子连接上。改变文本为密码。
③ 拖动一个 DB Source channel 到 Channel 视图中。双击打开属性窗口。作为‘Name in Quest3D’，使用‘MySQL’并选择‘Rename channel’选项。 单击‘OK’。
③ 将‘DB DriverMySQL’ channel 连接到‘DB Source’ channel。
③ 添加一个 DB Info Value 并将它连接到‘Database’ Channel Caller。双击 DB Info Value channel 打开属性窗口。 从下拉列表中选择‘Database Connected’选项。 从第二个下拉列表中选择‘MySQL’。注意连接的名称在 DB Source channel 中指定。单击‘OK’。

③ 将‘MySQL’ channel 连接到‘One Time’ channel。

③ 双击‘One Time’ channel 打开属性窗口。 单击‘Reset’按钮调用 channel 的子。 一段时间后，DB Info Value channel 的值应该变为 1。 关闭‘One Time’ channel 的属性窗口。

③ 拖动一个 DB Query channel 到 Channel 视图并连接到‘Retrieve Data’ channel。 双击‘DB Query’ channel 打开属性窗口。从第一个下拉列表中选择‘MySQL’。作为‘Name of this query’，使用‘Get Player Info’。 作为‘Actual query’，使用 SELECT * FROM player。这将从‘player’表中取出所有数据。 单击‘Execute’按钮，然后单击‘Results’按钮。 弹出显示结果对话框。 单击‘Close’，然后单击‘OK’。

③ 添加一个 DB Text channel 并将它连接到‘Text Out’ channel。 双击 DB Text channel 打开属性窗口。 选择‘Get Player Info’查询的‘name’字段。 选择‘Rename channel’选项并单击‘OK’。 注意名称显示在 Animation 3D View 窗口中。
拖动一个 Value channel 到 Channel 视图中并将它连接到 DB Text channel 的第一个子连接上。 重命名该 Value 为‘Index’。



③ 改变‘Index’ channel 的值为‘1’以获取‘name’列的第二个记录。 注意显示的名称改变了。

完成后的场景：

..\Tutorials\4.3 – Database connectivity\Datase connectivity 1 – Complete.cgr

保存当前工程。 下面的步骤将处理一个新的场景。

需要的 Quest3D 场景：

③ ..\Tutorials\4.3 – Database connectivity\Datase connectivity 2.cgr

需要的模版：

③ Variables \ Text \ Array Text (x2)

Step by step:

③ 打开文件，‘Database connectivity 2.cgr’。 它包含一个数据库连接逻辑，与上面的例

子相似。

③ 拖动一个 Array Text channel 到 Channel 视图中并将它连接到'Set Name Text' channel 的第二个子连接上。 双击'Array Text' channel 打开属性窗口。 从'Table'下拉列表中选择'Player'。 从'Column'下拉列表中选择'Name'。 选择'Update channel name'选项单击'OK'。

③ 右键单击'Player: Name' channel 并从菜单中选择'General Properties'。 设置'Update' to 'Continuous'并单击'OK'。

③ 创建'Current Row' channel 的快捷方式并将它作为索引连接到'Player: name' channel.

③ 切换到 Run Mode。

③ 单击'F2'将数据库数据导入到 Quest3D 中。

③ 切换到 Edit Mode。

③ 单击'Array Manager'标签并选择'Player'表。 注意'Name'列被来自于外部数据库的相应列的数据填充。

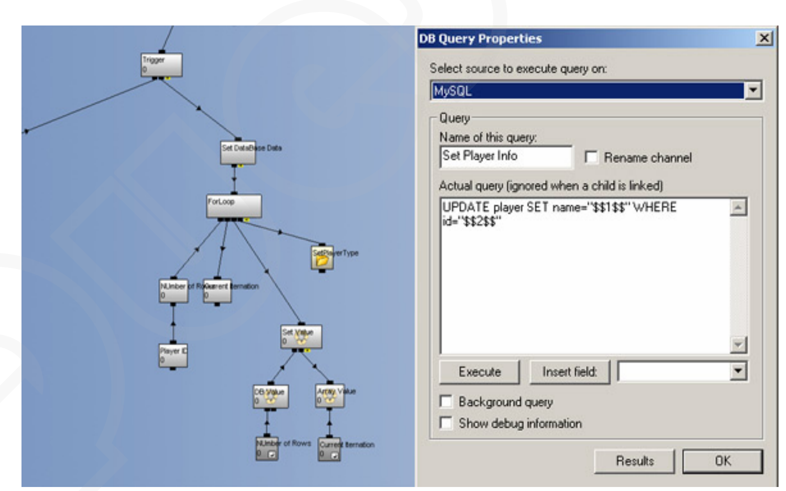
③ 单击'Channel Graph'标签。

③ 添加另一个 Array Text channel 并将它连接到'Set Player Name' channel 的第一个（空的）'Input'子连接上。 在'Player'表中设置数组为'Name'列的引用从而重命名它。

③ 右键单击新的'Player: Name' channel 并从菜单中选择'General Properties'。 设置'Update' to 'Continuous'并单击'OK'。

③ 创建 For Loop 中的 'Current Row' channel 的快捷方式，并将它作为索引连接到刚刚创建的'Player: Name' channel。

双击'Set Player Name' channel。 填写如下查询： UPDATE player SET name="$$1$$" WHERE id="$$2$$"这将替换远程数据库'player'表中的的所有'name'列。 'id'列作为主索引。 注意'$$1$$'作为'Set Player Name' channel 的第二个子连接。 '$$2$$'作为第三个子连接。 单击'OK'。



③ 单击'Array Manager'标签。

③ 改变'Player'表中'Name'列的一个名字。 注意 Animation 3D View 窗口右侧栏中的名称改变了，但是左侧栏没有。

③ 切换到 Run Mode。

③ 单击'F3'用 Quest3D 数组数据更新远程数据库。

③ 单击'F2'将远程数据库数据重新导入到 Quest3D 中。 注意 Animation 3D View 窗口左侧栏的被更新了。

③ 切换到 Edit Mode。

- ③ 双击‘Insert Player Name’ channel。 填写如下查询： INSERT player SET name="$$1$$"这将向远程数据库的‘player’表中添加一个新的记录。 注意‘$$1$$’作为‘Insert Data’ channel 的第二个子连接。 单击‘OK’。
- ③ 切换到 Run Mode。
- ③ 单击‘F4’添加新的记录到远程数据库中。
- ③ 单击‘F2’将远程数据库数据重新导入到 Quest3D 中。 注意 Animation 3D View 窗口左侧栏的被更新了。

完成后的场景：

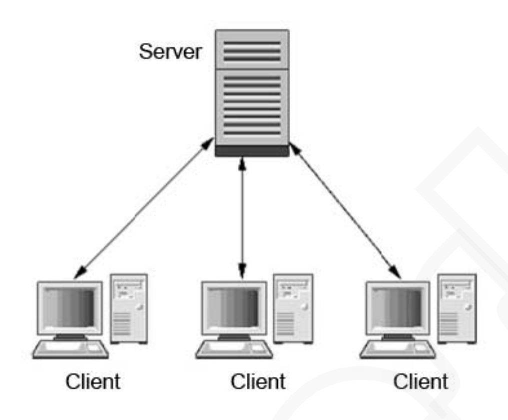
- ③ ..\Tutorials\4.3 – Database connectivity\Datase connectivity 2 – Complete.cgr

4.4 网络

某些应用可能为多用户设计。 例子包含一个虚拟聊天室，基于团队的训练模拟和在线多人游戏。 每一个用户必须运行一个程序的实例。 每一个实例必须持续的传输它的当前状态。 网络功能用来在计算机之间通信。

注意只有在 Quest3D 企业版和 VR 版中网络才是可用的。

在大多数情况下，网络中的一台计算机被用作‘server’（服务器）。 其他的计算机被称为‘clients’客户端。



服务器处理所有数据的传输。 它接受所有来自客户端的更新，也发送数据到每个客户端。 服务器只在客户端请求数据时发送。

Ques3D 网络应用被设置为常规的‘single user’工程。客户端载本地运行大多数应用。只有被其他用户影响的数据才会通过网络向服务器请求。 例子中包含用户位置，化身外观和聊天窗口中的文字。

客户端不需要安装 Quest3D。 客户端可以运行发布版，独立的‘.exe’文件或‘.q3d’ Viewer 文件。

通常情况下，服务器也是一个客户端。 如果没有这种情况，该计算机被称为‘dedicated server’（专用服务器）。

传输速率受限于服务器和客户端。 Quest3D 支持 32 个用户的网络应用。 更大的用户数量在技术上是可以的，但是可能导致性能的降低。

登录

客户端必须‘log in’（登录）服务器。 可以从所有运行网络应用的列表中选择服务器，局域网或是一个特定的网络地址。 产生或更新这样一个可用列表的网络程序被称为‘enumerating’（枚举）。

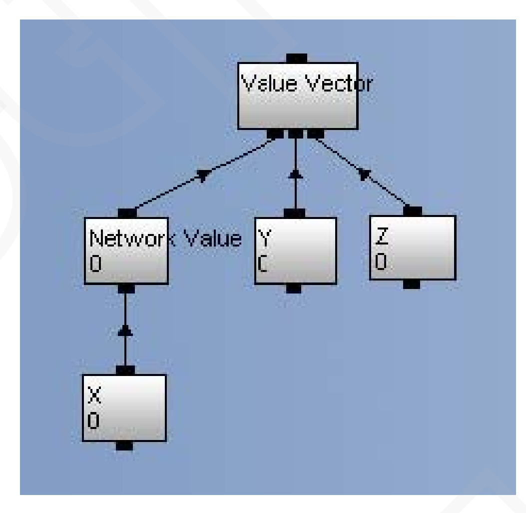
每个网络应用都有一个唯一的 ID 以确保客户端正确登录。

初始化

登陆后，客户端程序被初始化。 这个包含设置开始位置，改变外观和设置角色或化身类型。 此外，对这些元素的控制必须被分发到新登录的客户端。 用户对场景元素的控制可以通过动态改变的值来共享。

数据传输

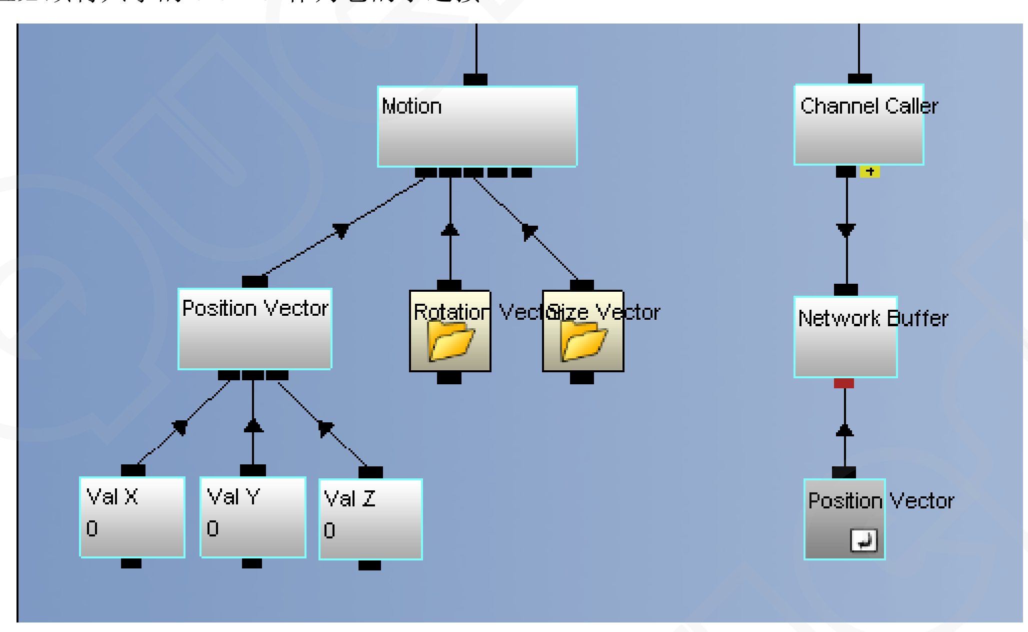
Quest3D 中的网络数据可以通过使用 Network Value, Network Matrix, Network Text 和 Network Buffer channel 来传输。 前三种类型可被插入到父和子之间。



Network Values, Network Matrices 和 Network Texts 可被发送和接收。 此外，可以设置传输确认，以确保数据正确地传送到目的地。 这种方式会比常规的传送方式慢。 最后，Network Value 和 Network Matrice 可被设置为‘Continuous’移动，以产生平滑动画。

缓存

其他类型的 Channel 可以通过使用 Network Buffer channel 来传输。它必须被独立调用，并且必须将共享的 channel 作为它的子连接。



网络行为

Network Actions channel 包含大量的功能例如创建和移除一个服务器，断开客户端和设

置用户控制。 它也包含大量的时间触发器例如创建服务器，客户端登录和拒绝连接。

网络信息

Network Info Value channel 包含大量的功能，例如连接状态，当前用户数和连接的用户总数。

Network Info Text channel 包含会话和计算机名。

网络触发器

网络触发器可以用来保证一个事件在所有的客户端上被触发。 常规的触发器不够精确而不能用于网络。

实例

网络功能运行两台或多台计算机之间相互通信。 Quest3D 中网络工程被设置为'但用户的'程序。 通常包含服务器和客户端的代码。

下面的实例将演示在 Quest3D 中连接到服务器。 第二个实例是一个高级例子，显示了用户对变量的控制。

需要的 Quest3D 场景：

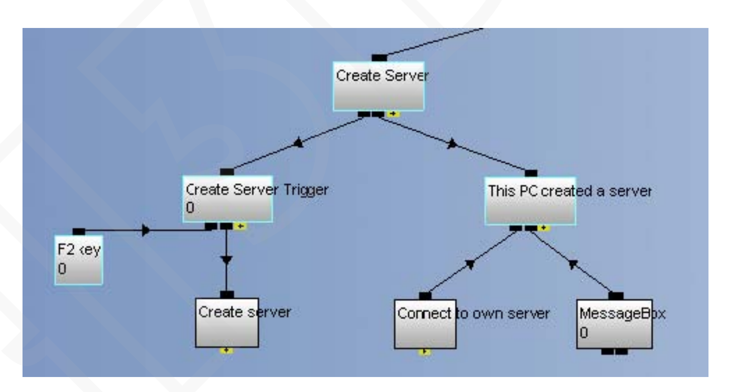
- ..\Tutorials\4.4 – Networking\Networking 1.cgr

需要的模版：

- Enterprise \ Network \ Network Actions (x3)
- Logic \ Expression Value
- Enterprise \ Network \ Network Info Value

Step by step:

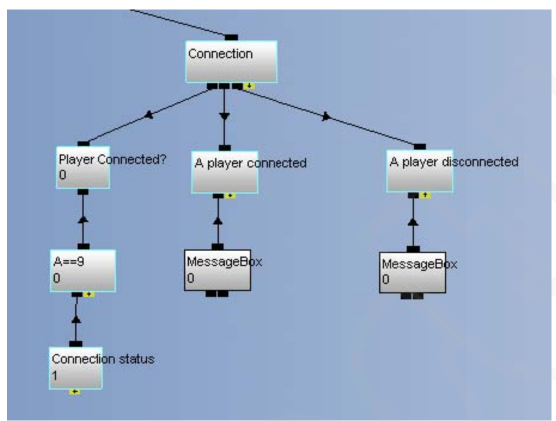
- 打开文件，'Database connectivity 1.cgr'。 它包含一个简单的逻辑结构。
- 拖动一个 Network Actions channel 到 Channel 视图中并将它连接到'Create Server Trigger' channel。 双击'Network Actions' channel 打开属性窗口。 从属性窗口的下拉列表中选择'Create Server'。 选择'Update channel name'选项单击'OK'。
- 添加另一个 Network Actions channel 并将它连接到'Create Server' channel。 设置并重命名该 Network Actions channel 为'This PC created a server'。
- 拖动另一个 Network Actions channel 到 Channel 视图中并将它连接到'Network Logic' channel。 设置并重命名该 Network Actions channel 为'Connect to own server'。
- 连接'MessageBox' channel 到'This PC created a server' channel。



③ 添加一个 Expression Value 并将它连接到‘Player Connected?’ channel。

③ 拖动一个 Network Info Value channel 到 Channel 视图中并将它连接到刚刚添加的 Expression Value。 双击 Network Info Value 并重命名为‘Connection Status’。

③ 双击 Expression Value 并改变公式为‘A==9’。 选择‘Rename Channel to formula’选项并单击‘OK’。 连接状态‘9’意味着一个连接被建立了。



③ 切换到 Run Mode。

③ 单击‘F2’。 一段时间后，建立了一个服务器并且‘Connect to own server’被触发。 一段时间后显示‘A player connected’信息。

③ 切换到 Edit Mode。

完成后的场景：

③ ..\Tutorials\4.4 – Networking\Networking 1 – Complete.cgr

保存当前工程。 下面的步骤将处理一个新的场景。

需要的 Quest3D 场景：

③ ..\Tutorials\4.4 – Networking\Networking 2.cgr

需要的模版：

③ Enterprise \ Arrays \ Array Channel

③ Enterprise \ Network \ Network Matrix

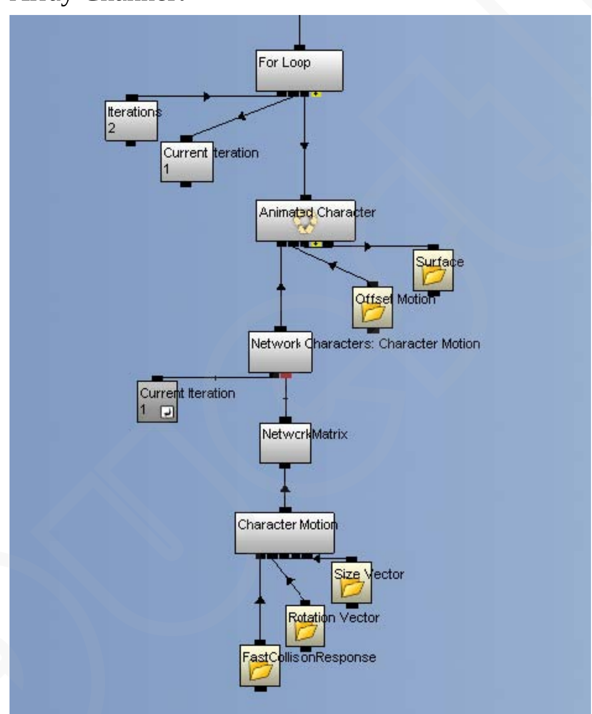
Step by step:

③ 打开文件‘..\Tutorials\Networking 2.cgr’。 它包含一个环境，一个角色和一个第三人称视角的行走像机，还有一个网络逻辑。

③ 进入 Animation Section。

③ 切换到 Run Mode。

③ 单击‘F2’创建一个服务器。 一段时间后，一个三维场景显示出来。 这意味着创建了一个服务器并且建立了一个客户连接。
③ 在场景中行走以测试场景。
③ 切换到 Edit Mode。
③ 进入 Channels Section。
③ 切换到 Edit Mode。
③ 进入 Channels Section。
③ 断开‘Animated Character’ channel 和‘Character Motion’ channel 之间的连接。
③ 添加一个 Array Channel channel。 从第一个下拉列表中选择‘Characters’表。 从第二个下拉列表中选择 ‘Character Motion Network’。 选择‘Update channel name’选项单击‘OK’。
③ 添加一个 Network Matrix channel 并将它连接到‘Characters: Character Motion Network’ Array Channel。
③ 将‘Character Motion’ channel 连接到‘Network Matrix’ channel。
③ 创建‘Current Iteration’ channel 的快捷方式并将它连接到‘Characters:Character Motion Network’ Array Channel 的第一个子连接上。 Character Motion Network’ Array Channel。



③ 进入 Animation Section。
③ 切换到 Run Mode。
③ 使用方向键和鼠标在场景中移动以测试场景。 单击‘1’到‘3’键切换纹理。

> 这个实例的下面部分需要第二台计算机作为客户端。 这两台计算机需要通过局域网相互连接。

③ 从 File 菜单中，选择‘Save As’。 保存当前工程到工程目录。
③ 从 File 菜单中，选择‘Publish’。 发布工程为独立的‘.exe’文件。
③ 拷贝‘.exe’文件到客户端。

- ③ 在客户端运行’exe’文件。
- ③ 在客户端，单击‘F4’键登录。 一段时间后，一个三维场景显示出来。
- ③ 在客户端，使用方向键和鼠标在场景中移动以测试场景。 在服务器端，注意客户端化身的角色在走动。
- ③ 在服务器端，单击‘1’到‘4’键改变服务器端角色的外形。 注意这种改变在客户端是实时显示的。



完成后的场景：

- ③ ..\Tutorials\4.4 – Networking\Networking 2 – Complete.cgr
- ③ ..\Tutorials\4.4 – Networking\Networking 2 – Complete.exe

4.5 Lua 脚本

Quest3D 的 channel 的构建模块包含预定义的 C++ 和 DirectX 代码。 整个 channel 集合为实时 3D 开发提供了许多功能。

尽管 channel 系统的开放性和逻辑结构，某些功能可以通过脚本语言更加有效和容易的处理。 Quest3D 指出 Lua，一个免费的第三方脚本环境。

Quest3D 中的 Lua 尤其可以用于加载或卸载 channel 组，复杂的计算和迭代器结构（‘for loops’）。

Lua channel 的脚本显示在它的属性窗口中。

Lua 功能

Lua channel 可以有两个函数中的一个或两个。

第一个是‘CallChannel’当 channel 被简单调用时被执行。

第二个是‘GetValue’，当 Lua channel 被用作一个数值时被执行。 Lua channel 的实际值等于脚本函数末尾的‘returns’的值。

变量

就像其他程序和脚本语言一样，Lua 脚本中使用的变量必须声明。 该语句为变量分配内存空间。‘local’意味着变量只会被使用在一个特定的地方，在这里是在 Lua channel 中。脚本执行完后，内存再次被清空。

本地变量

也可以在声明之上将一个值赋给一个变量。 local variable = 1

计算方式与其他程序语言的工作方式相同。 variable = variable + 1

组函数

Quest3D 支持如下标准的 Lua 组函数： ‘base’, ‘string’ 和 ‘math’ 此外，‘q’组包含 Quest3D 特定的功能。

两个实例函数是：

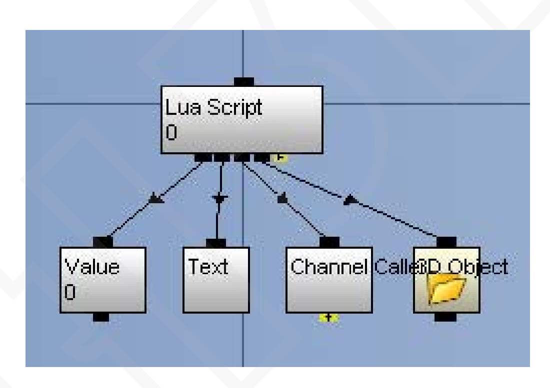
local variable = math.cos(value)

local variable = q.GetTick()

第一个例子计算括号中值得 cosine. 第二个例子从 Quest3D 的 Lua 组‘q’中取得 Tick Count

子 channel

Lua channel 可以连接任意数量和类型的子 channel。



在 Lua 中可以使用适当的函数可以访问它的子 channel。 同样，它们必须首先声明。 下面的语句访问连接到 Lua channel 的第一个子 channel（在位置‘0’）。

local variable = channel.GetChild(0)

子 channel 可以使用如下的语句来调用：

```
local variable = channel.GetChild(0)
variable:CallChannel()
```

注意使用预定的函数和本地结构的不同。 对于预定的结构例如‘channel’，一个‘.’在此函数之前。 (dot). 对于本地结构例如变量，一个‘:’在此函数之前。 (double colon punctuation mark).

连接到 Lua channel 的 Value channel 实际值可以通过如下语句访问：

```
local variable = channel.GetChild(0)
local value = variable:GetValue()
```

文本可以通过如下语句来取得：

```
local variable = channel.GetChild(0)
local text = variable:GetText()
```

子 channel 的值或文本可以使用如下语句来设置：

```
variable:SetValue(value)
variable:SetText(text)
```

For Loops

迭代器结构，在 Quest3D 中类似于‘For Loops’，可以使用如下语句来写：

```
while i < max do
    i = i + 1
end
```

加载组

尤其是用来加载和卸载组。 该语句是：

```
q.LoadChannelGroup("group.cgr", "PoolName", instance)
q.RemoveChannelGroup("PoolName", instance)
```

这些命令使得 Quest3D 中的组管理非常容易。

更多关于 Lua 的信息可以在其官方网站上找到。
http://www.lua.org

实例

Lua 扩展了 Quest3D channel 的功能。 下面的练习使你熟悉 Lua channel 的内部工作和它的两个主要功能。 ‘CallChannel’和 ‘GetValue’。

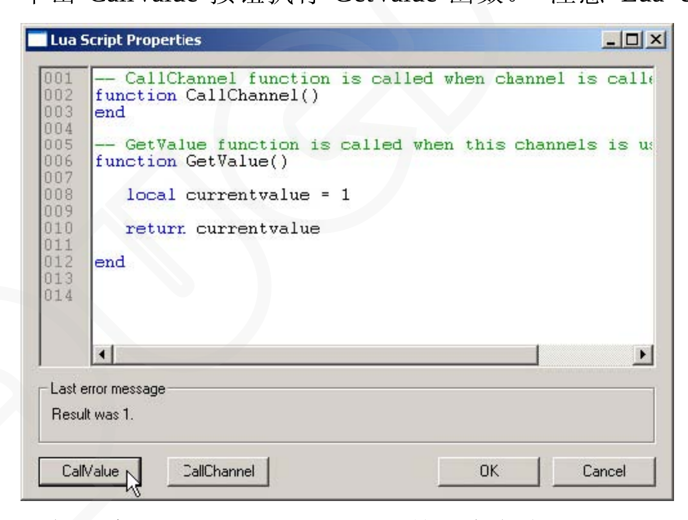
第二个实例详细的介绍 Quest3D 中 Lua 的高级使用——动态加载。

需要的模版：

- ③ Logic \ Empty Lua Script (x2)
- ③ Logic \ Channel Caller.
- ③ Variables \ Value \ Value (x2)
- ③ Logic \ User Input \ User Input

Step by step:

- ③ 从 File 菜单中选择‘New Project’开始一个新的 Quest3D 工程。
- ③ 拖动一个 Lua channel 到 Channel 视图中。 注意它被自动设置为 Start Channel。 双击打开属性窗口。 该窗口中已经包含两个函数的脚本。
- ③ 在‘GetValue’函数中，通过键入如下代码声明了一个新的变量。 local currentvalue = 1
- ③ 使用 return currentvalue 替换 return -1 来设置 Lua channel 的值为变量的值。 return -1 with return currentvalue
- ③ 单击‘CallValue’按钮执行‘GetValue’函数。 注意 Lua channel 的实际值变为‘1’。



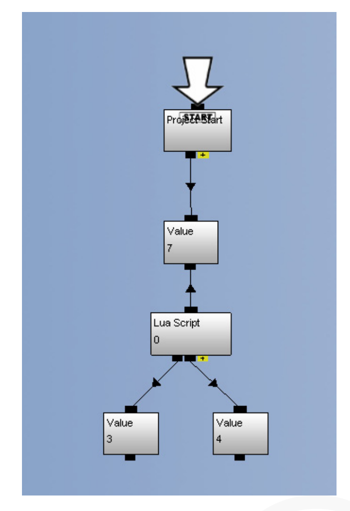
- ③ 添加一个 Channel Caller channel 并重命名为‘Project Start’。设置它为 Start Channel。
- ③ 添加一个 Value channel 并将它连接到‘Project Start’ channel。
- ③ 将 Lua channel 连接到 Value。现在 Lua channel 将作为一个值在每一帧中调用。换句话说，Lua 脚本的‘GetValue’函数将在每一帧中被执行。 或： ‘currentvalue’变量的值在每一帧中被计算并返回。
- ③ 如下改变脚本： function GetValue() local currentvalue = 1 currentvalue = currentvalue + 1 return currentvalue end 单击‘OK’。注意 Lua channel 的实际值现在为‘2’。
- ③ 拖动另一个 Value channel 到 Channel 视图中并将它连接到 Lua channel。 改变它的

值为‘3’。

③ 如下改变 Lua 脚本以访问它的子 channel。 function GetValue() local child0 = channel.GetChild(0) local currentvalue = child0:GetValue() return currentvalue end 单击‘OK’。 注意 Lua channel 的值现在与它的子 channel 的值相同：‘3’。 ‘3’.

③ 添加另一个 Value channel 并将它连接到 Lua channel 的第二个子连接上。改变它的值为‘4’。

③ 改变脚本以便返回两个子 channel 的和。可能有多个方案。下面是一种： function GetValue() local child0 = channel.GetChild(0) local value0 = child0:GetValue() local child1 = channel.GetChild(1) local value1 = child1:GetValue() local sum = value0 + value1 return sum end 单击‘OK’。 注意 Lua channel 的值为‘7’，为它两个子 channel 的和。



③ 对于简单的计算，Expression Value channel 更加易用。

③ 拖动另一个 Lua channel 到 Channel 视图中并将它连接到‘Project Start’ channel。

③ 添加一个 User Input channel 并将它连接到 Lua channel。

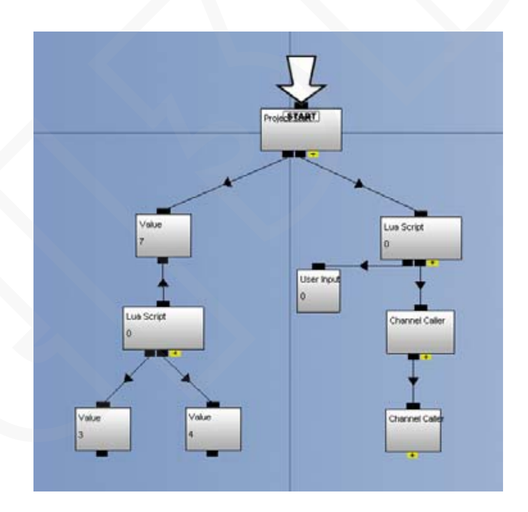
③ 拖动一个 Channel Caller 到 Channel 视图中并将它连接到 Lua channel 的第二个子连接上。

③ 添加另一个 Channel Caller 并将它连接到第一个 Channel Caller。

③ 改变‘CallChannel’函数脚本以便只有当第一个子等于‘1’时才会调用第二个子 channel。 与如下代码相似： function CallChannel() local child0 = channel.GetChild(0) local userinput = child0:GetValue() local child1 = channel.GetChild(1) if userinput == 1 then child1:CallChannel() end end 单击‘OK’。 注意第一个 Channel Caller 被激活。 这是由于脚本中的‘GetChild’语句。

③ 切换到 Run Mode。

③ 按住空格以测试脚本。 当空格按下时，第二个 Channel Caller 被调用。 释放空格。第二个 Channel Caller 不在本调用。



③ 对于简单的条件调用，If channel 更加易用。

完成后的场景：

③ ..\Tutorials\4.5 – Lua Scripting\Lua scripting 1 – Complete.cgr

保存当前工程。 下面的步骤将处理一个新的场景。

需要的 Quest3D 场景：

③ ..\Tutorials\4.5 – Lua Scripting\Lua scripting 2.cgr

Step by step:

③ 打开文件，‘Lua Scripting 2.cgr’。 它包含一个简单的逻辑结构。 注意蓝色的 Public Call Channel。 它报告一个错误：‘Not Found’。 这是正确的。

③ 双击 ‘Lua: Load Group’ channel 打开它的属性。 输入如下代码： function CallChannel() q.LoadChannelGroup (“Environment.cgr”, “Environment”, 0) end 该脚本加载 channel 组‘Environment.cgr’到‘Environment’池中。 零表示实例。 单击‘OK’。

③ 切换到 Run Mode。

③ 单击空格键加载特定的组。 注意环境显示在 Animation 3D View 中。

③ 切换到 Edit Mode。

③ 添加一行以加载组‘Character.cgr’到‘Character’池中。 function CallChannel() q.LoadChannelGroup (“Environment.cgr”, “Environment”, 0) q.LoadChannelGroup (“Character.cgr”, “Character”, 0) end

③ 切换到 Run Mode。

③ 单击空格键加载这两个特定的组。 注意角色和没有显示在 Animation 3D View 中。 这是因为它还没有被调用。

③ 切换到 Edit Mode。

③ 单击‘Groups’标签。

③ 拖动‘Character’池到 Channel 视图的下方。

③ 从下方的 Channel 视图中‘Character’ channel 拖动一个连接到上方的 Channel 视图中‘Render’ channel。 注意‘Character’ channel 变红了，并且在上方的 Channel 视图中创建了一个蓝色的 channel。

③ 单击‘Animation 3D View’标签。 注意角色显示在窗口中。

③ 双击蓝色的‘Character: Character -> Character’ channel 打开他的属性对话框。 选择‘Disable Autoload’选项单击‘OK’。

③ 打开‘Lua: Unload Group’ channel 并输入如下代码： function CallChannel() q.RemoveChannelGroup (“Character”, 0) end 该脚本卸载“Character”池。 零表示实例。 单击‘OK’。

③ 切换到 Run Mode。

③ 单击空格键卸载特定的组。 注意角色从 Animation 3D View 窗口中消失了。 同时注意蓝色的‘Character’ channel 报告一个‘Not Found’错误。

③ 单击空格键再次加载特定的组。 注意角色重新显示出来，并且蓝色的‘Character’ channel 再次可用。



完成后的场景：

③ ..\Tutorials\4.5 – Lua Scripting\Lua scripting 2 – Complete.cgr

附录

A1 从 Max 和 Maya 中导入

3D Studio Max

准备

- ③ 删除所有不必要的物体（辅助线等）
- ③ 选择所有剩下的物体
- ③ 将所有物体转化为‘Editable Mesh’（可编辑网格）
- ③ 在所有物体上应用‘ResetXForm’
- ③ 再次将所有物体转化为‘Editable Mesh’（可编辑网格）
- ③ 在某些情况下一些物体的法线会错位。
- ③ 将这些物体进行反转多边形操作（大多是被镜像的物体）
- ③ 如果有蒙皮物体，应使其堆栈中只有一个‘Edit Mesh’修改器和一个‘Physique’/’Skin’修改器。

建模

- ③ 焊接所有距离在 0.001 之内的点。
- ③ 确认所所有的物体都应用了平滑操作（即使不需要，所有物体必须被赋予一个平滑组）。 如果不手动应用它，而是依靠原始创建时的自动平滑，将不能确定能够正确地转化到 Quest3D 中。
- ③ 执行一个‘Attach List’将所有共享同一材质的物体变为一个（例如楼梯的一级） 这将使得在 Quest3D 中变更材质的操作更加容易。
- ③ 只要一个物体被镜像，很可能需要反转它的多边形。 这一点只有当应用‘ResetXForm’和‘EditMesh’修改器时比较明显（准备导出为.X）
- ③ 不要创建两个距离很近的两个相互叠加的面。 这将在 Quest3D 中引起 Z 缓存问题（当像机位于相互叠加的面的上方时将出现闪烁的现象）。
- ③ 注意在 Quest3D 中只有面和多边形是可见的。 线不能被导入。

材质

- ③ 不能有双面材质。 如果一个物体需要从正反两个方向观察，克隆该物体反转它的多边形并将它附在原始物体上。
- ③ 使用标准的 3DSMax 材质。
- ③ 设置为‘on’的‘Wire’材质在 Quest3D 中是没有用的。 如果要创建细小的物体（例如

绳或线），必须使用常用的几何体来创建它。

- ③ 只用使用 UVWmap 修改器或 UnwrapUVW 修改器的物体才能正确导出为.x。不要在才是编辑器中使用映射特性（缩放，裁剪等）因为它们不能导出。

贴图

- ③ 将所有用到的贴图放在同一个文件夹下。 将.X 导入 Quest3D 中，它将扫描文件夹找贴图，因此将.X 场景导出到你的贴图文件夹中。
- ③ 不要使用小于 8x8 像素的 JPG 贴图。 这将使 DirectX 崩溃。
- ③ 对于不透明的物体可以使用 JPG 贴图（或者 DDS 贴图，如果你的目标平台的显卡支持）。 贴图的大小必须是 2 的幂次，也就是说 8，16，32，64，128，256，512 像素都使用可用的大小。如果显卡允许你也可以使用 1024 和 2048 像素的贴图(大多数较新的显卡都支持它)。
- ③ 注意 512x512 的贴图和 1024x1024 的贴图是 1:4，因此相比前者后者将占用 4 倍的缓存。 如果使用的贴图不是 2 的幂次方，Quest3D 将自动将它向上转化（例如，一个 768x972 的贴图将被转化化为 1024x1024）
- ③ 对于透明物体你可以使用 TGA 贴图（带有透明通道的 32 位贴图）或者两个 JPG 贴图（这里第二个将包含透明通道信息）或者一个 PNG 文件。 注意使用 TGA 是更好的选择因为它支持 mipmap（不同于两个 JPG）
- ③ 对于透明物体，可以使用带有透明通道的 DDS。 如果预先在 Photoshop 中生成了一个 MIP 贴图，那么 Quest3D 工程的启动将很快。 通常，所有启用 mipmap 的贴图在应用程序启动前都会产生它们的 mipmap。 使用 DDS 你可以预先产生它。

Maya

Quest3D 包含用于 Maya6 以上的.x 导出器。有很多为 Maya 安装.x 导出器的方法。 可以使用下述方法或可以安装 Microsoft DirectX 9.0c SDK。

Microsoft DirectX 9.0c SDK 可以从下述页面下载： http://msdn.microsoft.com/directx/

建议使用微软的导出器，如果需要，按照如下步骤安装并使用 Maya5 导出器。 老版本也可以使用如下步骤。

- ③ 在开始菜单中选择‘Act3D/Quest3D/Extras/Exporters/Maya’。
- ③ 单击‘X exporter for Maya 5 (Vertex Color)’。
- ③ 一个消息提示你阅读‘readme.txt’。 单击 OK
- ③ 解压所有的文件到一个临时文件夹中，如‘C:\Temp’。 安装完成后可以删除这个目录。

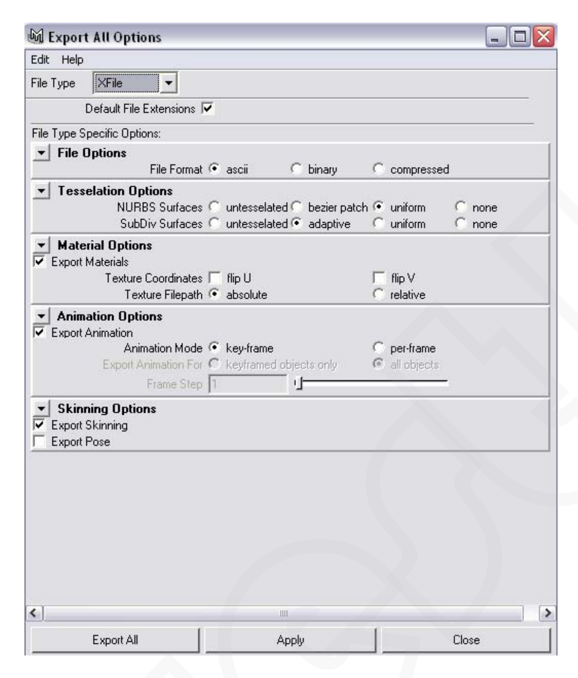
下面的建议来自于该 Readme.txt 并进过修正。

- ③ 确定 Maya 没有运行。
- ③ 拷贝‘xExport.mll’到 Maya 安装目录下的‘\bin\plug-ins’子文件夹中。
- ③ 拷贝‘xfileTranslatorOpts.mel’到‘\scripts\others’。
- ③ 拷贝‘bicubicBezierPatches.mel’到‘<mayapathname>\scripts\others’。

③ 拷贝‘msvci70.dll’和‘msvcr70.dll’到‘<WINDIR>/system32’。
③ 运行 Maya.
③ 从开始菜单中选择‘Window->Settings/Preferences->Plug-in Manager’。
③ 选择‘auto load’，靠近‘xExport.mll’
③ 选择‘loaded’，靠近‘xExport.mll’

按照如下步骤使用该导出器：

③ 创建一个物体或打开你的场景。
③ 选择 File -> Export All。 不要单击名称，而是单击选项按钮。
③ 显示导出选项。 从下拉列表中改变文件类型为“Xfile”。 将显示如下的选项。



③ File Option 设置文件构成的方式。 Ascii 工作的很好。
③ Tesselation Option 只有在你使用 Nurbs 或 Subdiv 时才有用。建议不使用这些设置，而是首先转化所有的物体为多边形。 如果用过常用的转化工具，那么结果将更加精确和可预测的。
③ Material Option 处理贴图。 可以反转 U/V 坐标，并设置相对或绝对贴图路径。 如果没有任何贴图，可以关闭“Export Materials”选项。
③ Animation Option 控制动画的导出方式。 如果只导出关键帧，Quest3D 将重新计算两帧之间的其他帧。 如果导出每一帧，文件将非常大但是更加精确。
③ Skinning option 在蒙皮角色上起作用。 如果没有使用骨骼，可以关闭“Export Skinning”选项。
③ 单击‘Export All’。 显示一个文件对话框。 指定文件名并单击 Export。 这将创建 x 文件，并可以将其导入到 Quest3D 中。

A2 用户接口（略）

A3 快捷键

F1	帮助
F2	Channels Section
F3	Animation Section
F4	Object Section
F5	Arrays Section
F6	Nature / Pathfinding Section
Control C	拷贝 channel
Control X	剪切 channel
Control V	粘贴 channel
`	创建 channel 快捷方式
Spacebar	创建/展开文件夹
Backspace	Level up
Pause Break	切换 Run / Edit 模式